الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR.YAHIA FARES DE MEDEA

FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département du Génie Électrique et de l'Informatique

Mémoire de Fin d'Études

Présenté pour l'obtention du Diplôme de Master en Génie Électrique

Option : Commande et électronique industrielle.

THEME

Réalisation d'un régulateur numérique PI pour commander un moteur à courant continu à base d'un microcontrôleur PIC16f877A.

Proposé et dirigé par,

Mr. BETTACHE Farouk

Préparé et présenté par :

Mr. LAIMECHE Salim

Mr. MOUHOUBI Farouk

AnnéeUniversitaire2015-2016

Remerciements

Nous Remercions dans un premier Temps, notre DIEU qui nous a donné la Volonté et le courage afin d'arriver à ce Modeste travail.

Nous remercions notre promoteur: Mr. F. BETTACHE qui nous ont orientés et aidé dans Toute la durée de notre travail par leurs critiques et leurs suggestions nombreuses.

Nous tenons à remercier tous les membres de jury pour avoir accepter de juger notre travail.

Nos sincères remerciements vont également à tous les enseignants du département de Génie électrique de l'Université de Médéa, et à tous les enseignants .

Dédicace

C'est avec une grande émotion, que je dédie ce modeste travail de fin d'étude à mes êtres les plus chers.
Mon cher amis farouk mouhoubi
Mes chers amis de medea city sans cité personne .

A mes très chers parents.
Au reste des membres de ma famille qui ont fait de moi ce que je suis aujourd'hui et qui ont veillé à guider mes pas durant toute ma vie par leurs aides, leurs grandes émotions et leurs sacrifices.

A tout ce qui m'ont aidé de près ou de loin pour la réalisation de ce travail.

A tous mes amis

A tous mes camarades durant les années anniversaires.

Dédicace

Je dédie ce modeste travail à tous ceux qui témoignent qu'il n'y a de DIEU, qu'ALLAH et MOHAMED est son envoyé et son prophète.

Mes très chères parents, pour leurs soutient moral et financier et d'être la lumière de ma vie, qui m'ont toujours encouragé pour terminer mes études, en leur espérant une longue vie et que Dieu les gardes.

- *A ma sœur ahlem et mes frères amine et hamza*
- *À mon binôme salim laimache*
- *A mes amies : amine, Dja fer, youcef, Mohamed Amine S, Mohamed Amine, aymane et tous les amies de luniversite de medea*

Mouhoubi farouk

الملخص

الهدف من عملنا هذا هو ادماج مصحح (معدل)PI داخل معالج PIC16F877A للتحكم في سرعة محرك ذو تيار مستمر (MCC)باستعمال لغة البرمجة MikroCلكتابة البرنامج الخاص لهذا التعديل وبرنامج Proteus Isisللمحاكاة قبل المرور الى التطبيق الذي قمنا فيه بتحويل العمل النظري اى عمل تطبيقي.

يتضمن العمل بعض المفاهيم الضرورية والاساسية حول محرك ذو تيار مستمر ' بالإضافة إلى لمحة عن المعالج pic16f877aونبض تعديل العرض(PWM) .

الكلمات المفتاحية: المعالج (pic16f877a)، الادماج، تصميم المحاكاة، **المنظم PI**،**MCC** محرك ذو تيار مستمر.

Résumé

L'objectif de notre travail consiste à implémenter l'algorithme PI dans un microcontrôleur PIC 16F877A pour la commande d'un moteur à courant continu en utilisant la programmation MikroC pour programmer notre PIC et le programme Proteus ISIS pour la simulation avant la fait en pratique.

Dans ce travail, quelques notions de base sur la machine à courant continu sont données.

Un aperçu sur le PIC 16F877A et modulation largeur impulsion(PWM) qui est dédié a cette application.

On programmant notre pic 16f877a a l'aide de logiciel WIN PIC 800 le programmateur

Mots clés : microcontrôleur **PIC : (Periphirol Interface contrôler)-** Modélisation-implémentation –Simulation **-** régulateur **PI -** moteur à courant continu**.**

Abstract

The aim of our work is to implement the PI algorithm in a 16F877A PIC microcontroller to control a DC motor using MikroC programming for programming our PIC and Proteus ISIS simulation program before done in practice.

In this work, some basics on the DC machine are given. An overview on the PIC 16F877A which is dedicated to this aplications.

And we realized these ideas by theoretical electronic components like digital and et pulse width modulation.

Is our peak 16F877A programming with the help of WIN 800 PIC programmer softwar**Keywords:** PIC microcontroller (**Periphirol Interface contrôler**)- Modeling –simulation **- PI** regulator- **DC** motor

SOMMAIRE

SOMMAIRE

Liste des figures…….…I

Liste des tableaux…….…..II

Liste des acronymes……...…III

Introduction générale…….. ..……..1

Chapitre I: la presentation de pic 16f877a et le MCC

I.1 Introduction……2

I.2LES CARACTERISTIQUESDU PIC16F877…….…… …….....2

I.3 DESCRIPTION GENERALDU PIC16F877…….....….....3

I.4Les timers. …….....9

I.4.1TMR0……..10

I.4.2TMR1……...15

I.5Les interruptions…….....…….....23

I.6.LES AVANTAGES DES MICROCONTROLEURS ……..…….36

I.7.CONCLUSION …….....36

I.8 .machines à courant continu ……....37

I.8 .1.introduction……..37

I.8.2. ORGANISATION DE LA MACHINE…….....37

I.8.9.VARIATION DE LA VITESSE DE LA MCC ……..…….45

I.8.10. PROCEDE DE COMMANDE DE MCC……..46

I.8.11.MODELISATION EN REGIME DYNAMIQUE……....47

I.8.13. *Encodeur incrémental*……...*50*

I.16. CONCLUSION…….....….….53

Chapitre II : La régulation analogique

II. Introduction…….…….….54

II.2 .Les actions PID ……...55

II.2.2.L'action dérivée ……...……57

II.3 .régulation analogique ……58

II.4. La conception d'un Compensateur PI Utilisation du Tuning Algorithme Ziegler-Nichols ……61

II.5. Régulation numérique ……...64

II.6.LA PARTIE DE SIMULATION ……..……..70

II.4. CONCLUSION……...75

Chapitre III : Simulation et Réalisation

III.1 Introduction…….……...76

III.2 Les outils de développements……76

III.*2.1.* Le proteus ISIS ……..….76

III.2.2 mikroC PRO……78

III.3.logiciel Win Pic 800……...….79

III.4.timer calculator…….....…80

III.5. simulation et réalisation…….……..81

III.5. 1.Introduction……...…...…..81

III.5.2.Organigramme ……82

III.5.3.Les étapes laréalisation……....…85

III.5.4. ETAPE 1 ……...…...86

III.5.4. 1.Configuration de PIC 16F877A ……...…86

III.5.4.2. configuration du convertisseur A/D ……..……...86

III.5.4.3.programme du convertisseur A/D sur mikro c……..87

III.5.4.4.éssaie pratique du convertisseur A/D ……89

III.5.5. ETAPE 2……...…...91

III.5.5.1.Interruption RB0 et RPM ……..….....91

III.5.5.2.Program de RB0 sur mikroc……...…..92

III.5.5.3. Essaie pratique Interruption RB0 ……...…94

III.5.6.ETAPE 3……...……95

III.5.6.1. Timer1 interrompre…….……..96

III.5.6.2.programme de TMR 1 en MIKROC……..96

III.5.6.3.essai pratique sur L'interruption TMR1……..…..…99

III.5 .7.l'encodeur……..101

III.5.7.1.SIMULATION ET ESSAIE PRATIQUE……..…*102*

III. 5.7.2.essaie pratique……..……...*103*

III .5.7.5. Mesure de la vitesse du moteur……..*103*

III.5.7.5.1. programme pour la vitesse sur MIKROC……..….……*122*

III.5.7.5.2. convertisseur numérique analogique …….….……...*107*

III.5.7.5.3.Programm du convertisseur numérique analogique MIKROC……...*108*

III.5.7.5.4. Essaie pratique du convertisseur numérique analogique DAC…….109

IV.5.8. le résultat final ……...……..……110

IV.5.8.1.Programmers final avec pid……..*111*

III.5.8.2.le schéma finale ……...…114

III.5.9. 1.DÉFINITION…….115

III.5.9. 2.FRÉQUENCE……....117

III.5.9. 3.Remarques…….…...….118

III.5.9. 4.RAPPORT CYCLIQUE…….….118

III.5.9.5. Program PWM sur MIKROC……..119

III.5.9.6. SIMULATION ET ESSAIE PRATIQUE PWM……121

III.5.9.7.Essai pratique PWM…….…….123

III.5.9.8. Résultat finale basé sur la PWM……123

III .5.9.10..Programme finale MIKROC……..…...124

III .5.10. Résultat de simulation ……..…127

III.5.11. réalisation final……..….132

III.12. CONCLUSION……..134

Conclusion générale…….

Annexe…….

Bibliographie .....

LISTE DES FIGURES

Figure I.1.*Boitier du PIC 16F877A*……..3

Figure I.2. *Architecture interne du PIC 16F877A* ……..…..…….……6

Figure .I.3. *Représente la minuterie TMR0*……*10*

Figure.I.4.**.** *OPTION_REGRegister*……..……..11

Figure I.5. *La fonction du PSA bit 0*……..*13*

Figure I.6.*La fonctiondu PSAbit1*…….....13

Figure I.7. *Minuteur TMR1*……..15

Figure I.8. *Minuteur TMR1 Vue d'ensemble* 16

Figure .I.9. *TMR1 Oscillateur*……..*17*

Figure I.10 *TMR1 en mode de minuterie*.....*18*

Figure I.11 *Minuteur TMR1 Oscillateur*.....*19*

Figure I.12. *Mode compteur* 20

Figure I.13.*T1CON Registre*.....*21*

Figure I.14. *Interrupt Registres système*.....*24*

Figure I.15. *Interrompre SFRs*.....*25*

Figure I.16. *INTCON Register*... 25

Figue. I.17. registre PIE1……..……27

Figure. I.18. PIE2 Register…….……27

Figure. I.19.PIR1 Register……..….29

Figure. I.20. registre PIR2……..30

Figure.I.21. registre 2-15 PCON…….31

Figure I.22. *Les différents PORT de PIC 16F877A*……..*32*

FigureI.23.*Schéma fonctionnel analogique numérique*.....34

Figure I.24.***a** : aiment, **b** : bobines* 38

Figure I.25. *Indui* …….*38*

Figure I.26. *Caractéristique magnétique d'une machine à courant con*……..…….…..39

Figure I.27. *Schéma équivalent du moteur à CC* ……..……..…… 40

Figure I.28. *Caractéristiques mécaniques et électriques* ……...…….…..... 41

Figure I. 29. *Constitution de la MCC* …….……..42

Figure I. 30. *Principe de fonctionnement* ….....43

Figure I.31. *Bilan des puissances d'un moteur à courant continu* ……....……44

Figure I. 32. *Bilan des puissances d'un générateur à courant continu* …….....45

Figure I. 33. ***a**. Variation de la vitesse en fonction de la tension d'induit **b**. Variation de la vitesse en fonction du flux d'inducteur* …….....48

Figure I.34. *Classification des huit fonctions de transfert du moteur à courant continu (obtenus en négligeant L, f ou C0).* ……...……...50

Figure I.35. *Encodeur incrémental*.....*52*

Figure I.36. *Encodage X2*.....*52*

Chapitre II

FigureII.1.*Schémade principe d'une chaine de régulation* ……..……..... 54

FigureII.2.*Schém asynoptique d'un régulateur PID* …….…….... 55

FigureII.3.*Schéma bloc du modèle watertank_comp_design* …….……...……...… 61

FigureII.4.*la vitesse de 3000 tr/min en fonction de temps* ……..…..…62

FigureII.5.*la vitesse de 3000 tr/min en fonction de temps* …….……...……..……63

FigureII.5.*la vitesse de 3000 tr/min en fonction de temps* ……..……64

Figure II.6. *Schéma blocs du compensateur PI* ……...…….... 66

Figure II.7. *la vitesse en fonction de temps* ……..….....67

Figure II.8. *Approximation rectangulaire* ……..…….....68

FigureII.9.*Approximation rectangulaire devancée* …..... 69

Figure II.10. *Approximation trapézoïdale* ……..……...….....70

FigureII.11.*Schéma bloc en z du compensateur PI.* ..…….....71

FigureII.12.*chéma bloc globale de régulation de MCC par PI en Z* 72

Figure. II.13. Schema *bloc MCC*……..…..*73*

Figure. II.14. *Schéma bloc de la régulateur PI et les gains*……..…..*73*

Figure. II.15. *la vitesse de 3000 tr/min en fonction de temps*……..……..*74*

Figure. II.16. *la vitesse de 3000 tr/min en fonction de temps*……..……*74*

Figure. II.17. *la vitesse de 3000 tr/min en fonction de temps*……..……..*75*

Chapitre III

Figure III.1. *Interface utilisateur ISIS* ……..……....77

Figure III.2. *Configuration du programmateur géré par Winpic800*……..…..*79*

Figure III.3. *Les étapes de transfert d'un programme*……..*80*

Figure III.4.Schéma synoptique de la chaine de régulation de la vitesse de MCC…….....81

Figure III.5.*Le schéma global sous logiciel proteus configuré par le DAC*……..*82*

Figure III.6. *Représente la simulation sous proteus configuré par PWM*……..*82*

Figure III.7.*organigramme de programme principale* ……..…….*83*

.

Figure III.8. *Organigramme de Programme De comptage de nombre de tours*……..*83*

Figure III.9.*Organigramme de Programme de la génération de la commande*……..*84*

Figure III.10. *L'entrée d'ADC RA0*……..…..*86*

Figure III.11 *.la conversion analogique numérique pour 2.5 v ……..……..…88*

Figure III.12. La conversion analogique numérique pour 5v……..…..89

Figure III.13. test d'ADC pour 2v …….……....90

Figure III.14. test d'ADC pour 4.81v 90

Figure III.15. Test de ADC pour 3.49v……...91

Figure III.16.test par simulation pour Vref=1.25v…….……94

Figure III.17.test d'une interruption RB0……...95

Figure III.18. *Schéma électrique du timer1……..99*

Figure III.19. Test de TMR1 ……*.99*

Figure III.20.l'affichage de TMR1 sur LCD……..100

Figure III.21.l'affichage TMR1 sur l'oscilloscope……100

Figure III.22.*Schéma de simulation d'un encodeur 200 révolutions……....101*

Figure III.23.l'affichage des comptes sur l'oscilloscope……102

Figure III.24.*Schéma de simulation de calcul de la vitesse……..106*

Figure III.25.Etage de puissance ……*109*

Figure III.26.*Schéma de DAC relié avec ampliope LF351*…….…..109

Figure III.27.Resultats de la realisation ……110

Figure III.28.resultats de la realisation …….....110

Figure III.29.*test de simulation pour Vref=1.25v……..115*

Figure III.30.*rapport cyclique ……..119*

Figure III.31.*Schéma globale de PWM sur le proteus……...121*
..
Figure III.32.*Schéma de la simulation pour* PWM 255bits……..121
…

Figure III.33.*Schéma de la simulation pour* PWM 128bits……...…122

Figure III.34.*test de la simulation pour* PWM 64 bits……..…122

Figure III.35.test de PWM pour 256 bits ……..*123*

Figure III.36.test de PWM pour 128 bits ……..*123*

Figure III.37.test de PWM pour 64 bits ……..*124*

Figure III.38. Test de simulation pour1.25 v…….128

Figure III.39. L'affichage sur l'oscilloscope……..…129

Figure III.40.test de la simulation pour 2.5v ……*129*

Figure III.41.l'affichage sur l'oscilloscope……..…….130

Figure III.42.test de simulation pour 5v ……*131*

Figure III.43.l'affichage sur l'oscilloscope……..…….132

Figure III.44. Schéma de réalisation final……132

.

Figure III.45.test et affichage ……..…*133*

Figure III.46.test de réalisation et affichage oscilloscope pour Vref=2.5v……....133

.

Figure III.47.test de réalisation et affichage oscilloscope pour Vref=5v…….134

LISTE DES TABLEAUX

Tableau I.1. *Broches du PIC16f877A.* ……...4

Tableau.I.2. Prescaler Taux de prediviseur……...12

Tableau I.3. *La valeur recommandées de condensateurs en fonction de l'oscillateur à quartz*……17

Tableau I.4. *PrescalerTaux*……23

Tableau I.5. *Bits de registre ADCON1*……..*36*

Tableau I.6. *Bits de registre ADCON0* ……...36

LISTE DES ACRONYMES

MCC: machine à courant continu.

PI: Proportionnelle intégral.

DC: Courant Continu (Direct Curent).

PWM : pulse width modulation (largeur d'impulsion modulation).

ADC : convertisseur analogique numérique.

DAC : convertisseur numérique analogique.

PIC: (Perifirol Interface Controler)

RPM : round per minute

Introduction générale

Introduction générale

L'application de la régulation de vitesse des moteurs électriques est très répondue dans les

Systèmes industriels et automatiques, l'un des critères d'efficacité d'une telle régulation est sa Robustesse face aux changements. Elle permet d'effectuer une régulation des paramètres Physiques (vitesse, position, température… etc.) des systèmes en boucle fermée, c'est à dire Un système capable d'atteindre et de maintenir une certaine consigne en utilisant une mesure, Dont il reste à déterminer la fonction permettant de corriger la commande en fonction de la Consigne initiale et de l'erreur mesurée.

Notre objectif dans ce mémoire est l'implémentation le régulateur PI dans le microcontrôleur Pic 16f877a pour faire la commande de moteur a courant continu pour ce travail on doit utiliser le logiciel MikroC pro pour la programmation du Pic et le logiciel Proteus ISIS pour la simulation, on va réaliser trois chapitres et une conclusion générale:

Le premier chapitre on a présenté le microcontrôleur Pic 16f877A et son architecture (interne et externe), leur caractéristique et ses différents registres.
Et les caractéristiques de moteur à courant continu.

Le deuxième chapitre contient deux parties, dans la première partie on a fait une étude théorique sur le régulateur PI et sur l'encodeur incrémental, dans la deuxième partie on a fait une régulation analogique et numérique avec le compensateur PI dont l'outil de simulation est le logiciel MATLAB Simulink.

Le troisième chapitre est consacré à l'implantation et à la simulation de la commande de moteur à courant continu et faire les résultats de simulation.

Nous terminons notre travail par une conclusion générale.

Chapitre I :

La présentation de PIC16F877A
Et le MCC

I.1 INTRODUCTION

L'objectif principal de ce chapitre est la présentation du cœur de notre application à savoir le microcontrôleur PIC16F877A.
L'architecture externe (brochage), l'architecture interne (registre interne), configuration des registres et l'interruption sont autant de paramètres auxquels nous y aborderons dans ce chapitre.

Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile, etc.

Des nombreux fabricants se sont implantés sur le marché des microcontrôleurs, la société américaine Micro-chip a mis au point un microcontrôleur CMOS, « Complementry Metal Oxide Semi-conductor » appelé PIC ; cette dénomination est sous copyright de Micro-chip, donc les autres fabricants ont été dans l'impossibilité d'utiliser ce terme pour leur propre microcontrôleur. Actuellement ce microcontrôleur est très utilisé.

I.2 LES CARACTERISTIQUES DU PIC 16F877A

Nous avons utilisé le microcontrôleur PIC 16F877A pour contrôler la vitesse du moteur, ce µc' est un circuit intégré à 40 broches dont les caractéristiques générales peuvent être résumé comme suit :

Vitesse d'exécution allant jusqu'à 20MHz permettant l'exécution rapide du programme, une instruction en cycle d'horloge de 200ns.

Une mémoire vive « RAM » de 368 octets, répartie sur 4 banque 0, 1, 2, 3 « Mémoire de données ».

Une mémoire EEPROM pour sauver des paramètres de 256 octets « mémoire de données ».

Une mémoire morte « ROM » mémoire programme de type FLASH de 8 kmots « 1mot=14bit ».

- 33 Entrées/Sorties programmables multiplexés.
- 8 entrées « canaux » du module de conversion analogique numérique 10 bits.
- 2 Comparateurs analogiques programmables avec une référence de tension, programmable également.
- Conservation des informations en mémoire vive jusqu'à 1.5v.
- Une plage d'alimentation de 2.0V à 5V.

- Module de capture PWM programmable permettant de capturer un signal « Logique » en fonction du temps, ou d'offrir un signal PWM : Module de Largeur d'Impulsion.
- De nouvelles fonctionnalités, comme les gestions de ports «Série».

Le PIC 16F877A parmi les plus puissants microcontrôleurs de la gamme « Mid-range » chez Micro-chip. Dans notre travail, on s'intéresse au convertisseur analogique/numérique au Timer1 pour la commande du moteur à courant continu. [1]

III.3. DESCRIPTION GENERALE DU PIC 16F877A

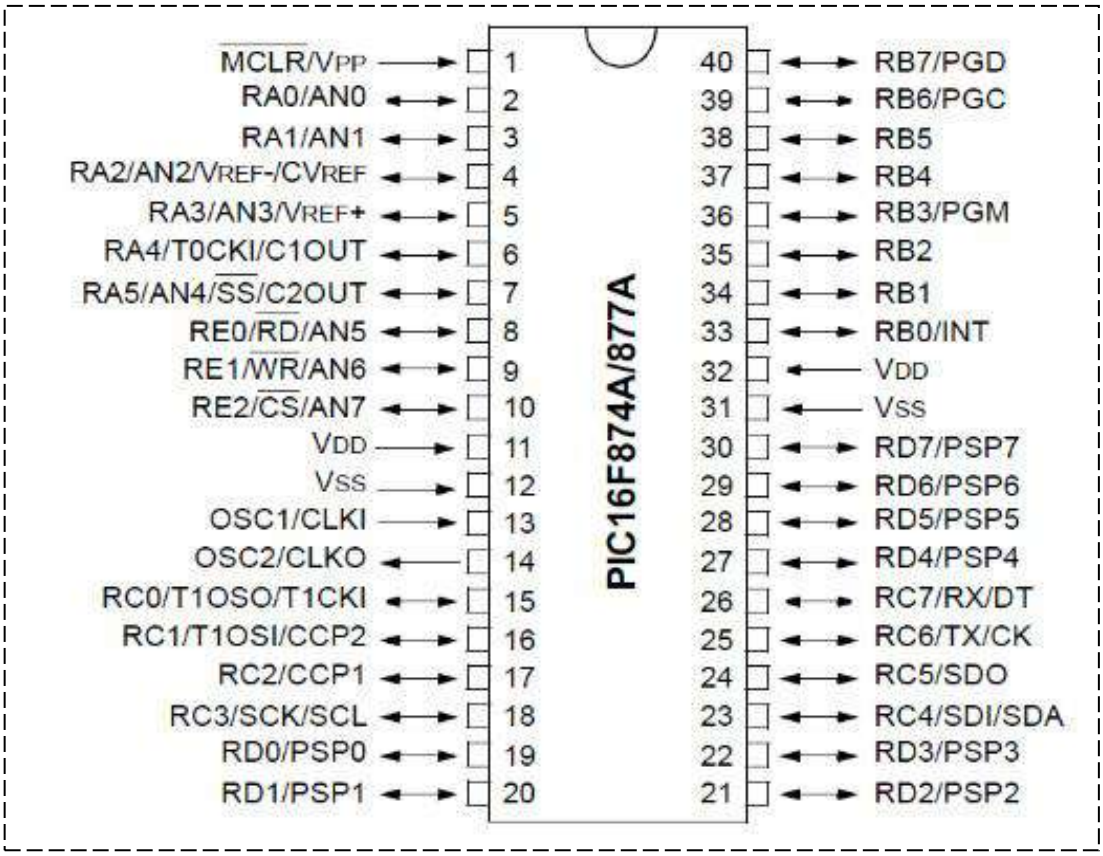


Figure I.1 : *boitier du PIC 16F877A* [1]

I.3.1. Architecture externe

Le microcontrôleur 16F877A fait partie de la sous famille des 16F87X. Cette branche fait partie intégrant de la famille des Pics Mid-Range, Le PIC 16F877A se présente sous la forme d'un boitier PDIP 40 broches, Le numéro peut être suivi d'un «A», et d'un «-XX» qui Donne la fréquence d'horloge maximum du composant. A l'heur actuel,

la version la plus courante est la version -20. Donc la fréquence maximale sera de 20MHz. [1]

PIN	UTILISATION
RA0 à RA5	Lignes d'entrées /sorties du port A
RB0 à RB7	Lignes d'entrées /sorties du port B
RC0 à RC7	Lignes d'entrées /sorties du port C
RD0 à RD7	Lignes d'entrées /sorties du port D
RE0 à RE3	Lignes d'entrées /sorties du port E
VDD	Broche d'alimentation +5V
VSS	Broche du 0V
MCLR	Reset du circuit
OSC1 et OSC2	Broche recevant quartz externe.

Tableau I.1. *Broches du PIC16F877A.*

a. Alimentation « VDD ET VSS »

Ce sont les broches d'alimentation du circuit, les tensions qui peuvent être appliqués sont :

- De 4V à 6V pour la gamme standard.
- De 2V à 6V pour la gamme étendue.

L'intensité du courant consommé peut aller de 1µA à 10mA.

La consommation sera en fonction de :

- La tension d'alimentation.
- La fréquence interne.
- Le mode de fonctionnement.

b. L'horloge

Deux pates 13et 14sont présentées sur le boitier du PIC 16F877A. La première nommée OSC1/CLKIN et la deuxième OSC/CLKOUT. Ces deux broches peuvent être utilisées de plusieurs manières; cela dépend de l'oscillateur ou de l'horloge employés:

XT : version à oscillateur à quartz «standard» fonctionnant jusqu'à une fréquence maximale de 4MHz

HS : High Speed, version à oscillateur à quartz capable de fonctionner jusqu'à une fréquence maximale de 20MHz.

RC : capable de fonctionner jusqu'à 4MHz mais avec une stabilité de fréquence plus faible que les versions à quartz.

LP « Low Power » : version à quartz également, mais prévue spécialement pour les applications à très faible consommation. C'est donc une version dont la fréquence maximale de fonctionnement est limitée à 20KHz.

c. MCLR

Cette broche sert à initialiser le PIC, ce dernier dispos de plusieurs sources de RESET. POR « Power on reset » : mise sous tension. Un front descendant sur MCLR déclenche l'initialisation du µc. Cette bronche peut être simplement reliée à VDD si on n'a pas besoin de RESET externe.

EXTERNAL RESET « mis à l'état de MCLR ».Remise à zéro extérieur. Il faut appliquer un niveau bas sur l'entrée.

RESET au moins 2µs pour que l'initialisation soit prise en compte.

WDT : chien de garde. Si le WDT arrive à la fin du temps de garde sans être rafraîchi il aura une initialisation du µc

BOR : baisse de l'alimentation. Si l'alimentation VDD chute en dessous de 4V pendant 100µs au moins, le microcontrôleur peut générer un RESET. [1]

I.3.2. Architecture interne

Cette architecture nous présente les principaux organes qui composent le µc, la figure (I.2) illustre cette architecture :

- La mémoire FLASH de 8K mots à 14 bits.
- La mémoire de donnée EEPROM à 256 octets.
- La mémoire de donnée RAM à 368 octets.
- Les registres particuliers W « registre de travail »; FSR et STATUS « d'état ».
- L'unité arithmétique et logique « UAL ».
- Quatre ports bidirectionnels E/S A, B, C, D.
- Huit niveaux de pile. [2]

Cependant, même si les op-codes sont codés sur 14 bits, les données et les résultats sont seront toujours sur 8 Bits.

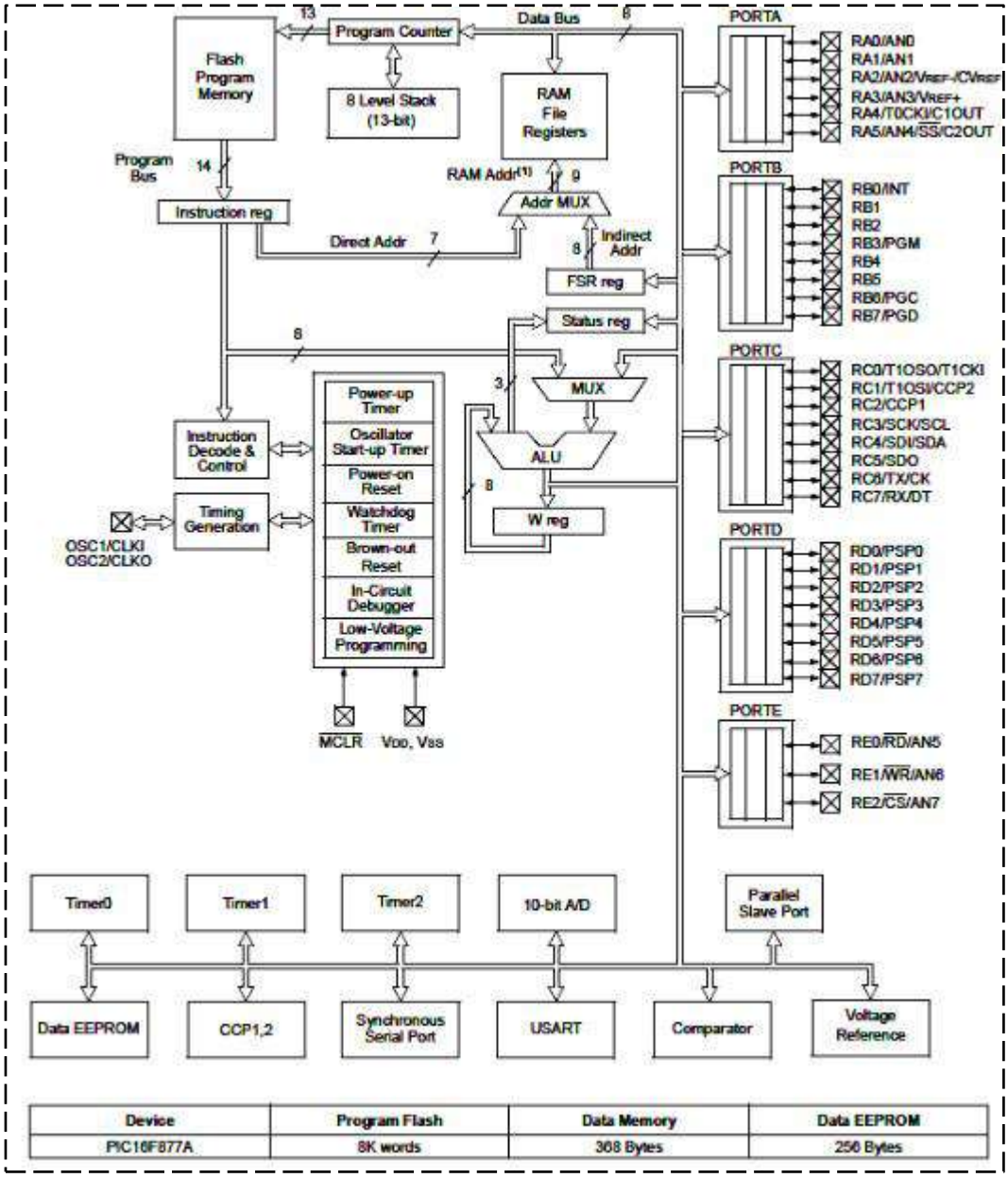


Figure I.2. *Architecture interne du PIC 16F877A*

I.3.3. Organisation de la mémoire

La mémoire du PIC 16F877A est divisée en 3 parties :

1. La mémoire programme :

La mémoire programme est constituée de 8k mots de 14 bits. C'est dans cette zone que nous allons écrire notre programme. Le PIC exécute une à une les instructions logées dans la mémoire de programme.

2. La mémoire EEPROM :

La mémoire EEPROM « Electric al Ecrasable Programmable Read Onlay Memory »,

est constitués de 256 octets, ces octets sont conservés après une coupure de courant et sont très utiles pour conserver des paramètres semi-permanents.

I.3.4. La mémoire RAM et organisation

La mémoire RAM est celle qui est sauvant utilisée. Toutes les données qui y sont stockées sont perdues lors d'une coupure de courant. La mémoire RAM disponible du 16F877 est de 368 octets. Elle est répartie de la manière suivante :

- 80 octets en banque 0, adresses 0*20 à 0*6F
- 80 octets en banque 1, adresses 0*A0 à 0*EF
- 96 octets en banque 2, adresses 0*110 à 0*16F
- 96 octets en banque 3, adresses 0*190 à 0*1EF
- 16 octets commun aux 4 banques, soit 0*70 à 0*7F = 0*F0 à 0*FF.

a. Plan mémoire pour les instructions « code programme »

Le plan mémoire est linéaire les adresses vont de 0000h à 1FFFh « 8k mots de 14 bits », Par page de 2k mots.

Les Pics n'ont qu'un seul vecteur d'interruption en 0004h. Lors d'une interruption, le sous-programme associé devra déterminer quel périphérique a demandé en mémoire de donnée comme avec les microcontrôleurs classiques, mais dans la mémoire programme. Elles sont utilisées lors d'appels de sous programmes, on ne peut pas imbriquer plus de 8 sous programmes.

b. Plan mémoire pour les données et registres interne
« SFR : Spécial fonction registre »

Le plan mémoire des données et des registres internes est découpé en 4 zones ou Bank de 128 octets, pour accéder à une zone il faut positionner les bits RB0 « bit 5 » et RP1 « bit 6 » du registre STATUS.

Les registres appelés General propose Registrer ne sont ni plus ni moins que des cases mémoires pour stocker les données. [2]

I.3.5. Les registres du PIC16F877A

La mémoire RAM est organisée en 4 banques pour le 16F8777A. La RAM est subdivisée de plus en quatre parties. Dans chacune des banques on trouve des «cases mémoires spéciales» appelées REGISTRES SPECIAUX et des cases mémoires «Libres» dont nous pouvons nous en servir à notre guise.

a. Le registre statu d'adresse 0X03 ou registre d'états

Comme son nom l'indique ce registre contient un certain nombre de bits d'états de l'unité centrale mais aussi des bits de sélection de la page qui s'appellent ici RP1 et RP0. De ce fait, il peut être lu comme n'importe quel autre registre étant entendu que certains bits d'états restant non modifiables.

b. Le registre FSR d'adresse 0X04

Le PIC 16F877A possède une unité arithmétique et logique « UAL » et un registre de travail W. L'UAL effectue des fonctions arithmétiques booléennes entre les données se trouvant dans le registre de travail et n'importe quel registre. Pour les instructions à deux opérandes, typiquement une donnée est dans le registre de travail « W » et l'autre opérande dans un autre registre ou se trouve une valeur immédiate constante. Pour les instructions à un seul opérande, la donnée se trouve soit dans le registre de travail « W » soit dans un autre registre.

Le registre de travail « W » possède 14 bits, utilisé pour les opérations avec l'UAL, ce n'est pas un registre adressable. Elle « UAL » est composée :

- D'un accumulateur 8 bits W : WORKING « travail », c'est lui qui effectue toutes les opérations arithmétiques et logiques.
- Un registre d'état 8 bits STATUS.

IRP	RP1	RP0	TO	PD	Z	DC	C
Bit7							Bit0

- Bit 0 ou bit C Carry c'est le bit de retenue en addition ou soustraction sur un octet. Il positionné a 1 si une retenue set générée depuis le bit de poids fort
- Bit 1 ou bit DC comme Digit Carry0 il est positionné à 1 si une retenue est générée du bit 3 « Bit de poids fort du quartet inférieur » vers le bit 0 du quartet supérieur. Il set utile pour corriger le résultat d'opérations effectuées en code BCD

- Bit 2 ou Z comme Zéro, ce bit est mis à 1 si le résultat de l'opération arithmétique ou logique est nul.
- Bit 3 ou PD comme Power Dowen, ce bit à 1 lors de la mise sous tension du circuit ou lors de l'exécution d'une instruction CLWDT relative au timer chien de garde il est mis à 0 par l'instruction SLEEP.
- Bit 4 ou bit TO comme Time Out, ce bit est à 1 lors de la mise sous tension du circuit ou lors de l'exécution d'une instruction CLWDT ou SLEEP, il passe à 0 si le timer chien de garde déborde.
- Bit 5 ou bit RP0 « Registrer Bank Select0 », ce bit sert à sélectionner l'une des deux pages de la mémoire « Page 0 ou Page 1 » sachant que chaque page mesure 128 octet. RP0 = 0 sélectionne la page mémoire 0 « adresse de 00 à 7F »

RP0 = 1 sélectionne la page mémoire 1 « adresse de 80 à FF »

- Bit 6 ou bit RP1 « Register Bank Select1 », permet la sélection des pages 2 et 3.
- Bit 7 ou bit IRP ce bit est prévu pour un future adressage paginé indirect. [1]

I.4. Les timers

Un **timer** est un périphérique matériel permettant de mesurer des durées (généralement inclus dans les microcontrôleurs). Son rôle est de permettre la synchronisation des opérations que le microcontrôleur est chargé d'effectuer.

Pour compter le temps ou des événements, les micro-contrôleurs disposent de blocs spécialisés, des timers. Un timer est constitué d'un compteur dont on peut ajuster l'horloge, lire la valeur et écrire la valeur. Les timers sont capables de générer des interruptions mais nous ne verrons pas cet aspect-là dans ce cours.

Les timers du microcontrôleur PIC16F887 peuvent être décrits brièvement dans une seule phrase. Il y a trois complètement
compteurs indépendants / compteurs marqués comme TMR0, TMR1 et TMR2. Mais ce n'est pas aussi simple que cela.

I.4.1.TMR0

La minuterie TMR0 dispose d'un large éventail d'applications dans la pratique. Rare de programmes ne l'utilisent pas, en quelque sorte. C'est très pratique et facile à utiliser pour écrire des programmes ou des sous-programmes pour générer des impulsions d'une durée arbitraire, le temps mesure ou de comptage des impulsions externes (événements) avec presque pas de limites.

Le module TMR0 de minuterie est un 8-bit temporisateur / compteur avec les caractéristiques suivantes:

l8-bit temporisateur / compteur;

8-bit pré diviseur (partagé avec Watchdog timer);

la source d'horloge interne ou externe programmable;

Interrompre en cas de débordement; et sélection de bord d'horloge externe programmable.

Figure I-3 ci-dessous représente la minuterie TMR0 schéma avec tous les bits qui déterminent son fonctionnement. Ces bits sont stockés dans le registre OPTION_REG. [3]

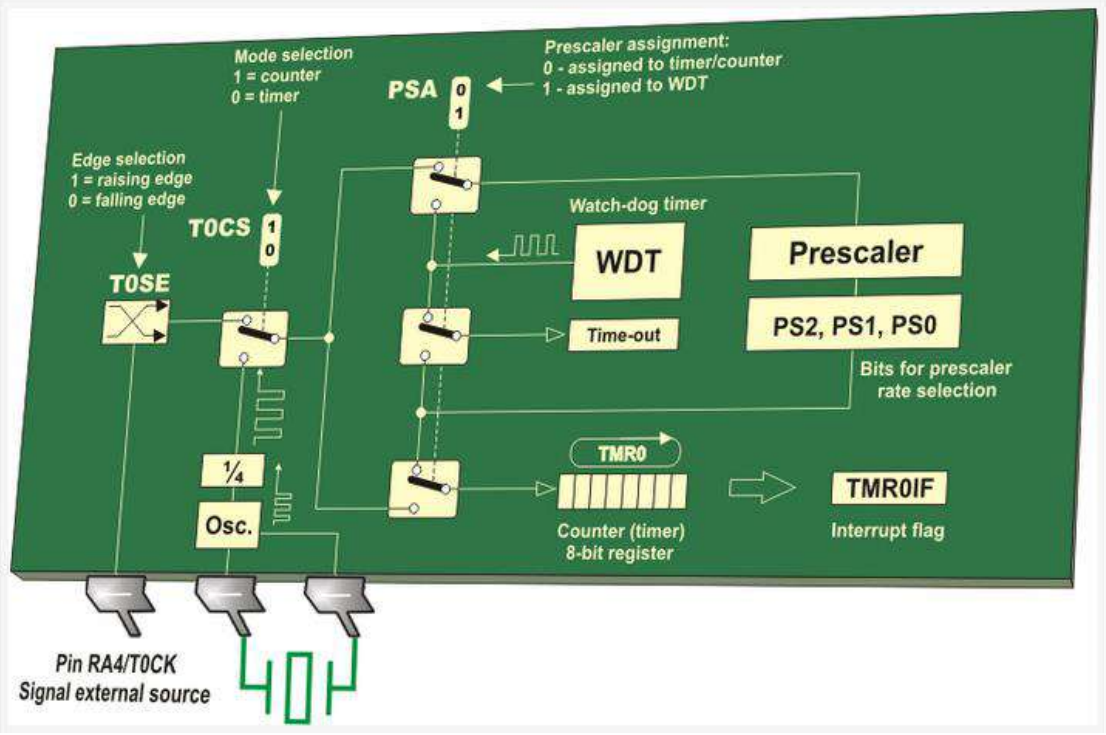


Figure .I.3. *Représente la minuterie TMR0. [3]*

OPTION_REG Register



Figure .I.3. *Représente la minuterie TMR0. [3]*

Le compteur/Timer TMR0 a les caractéristiques suivantes :

- Compteur sur 8 bits.
- Lecture / écriture de TMR0.
- Prédiviseur 8 bits programmable.
- Choix de l'horloge : interne en Timer et externe en compteur.
- Interruption au débordement (passage de FF à 00).
- Choix du front de l'horloge en mode horloge externe.

Tous les bits de configuration de TMR0 sont dans le registre OPTION en h'81' page 1 ou en h'181' page 3.

Le registre TMR0 est à l'adresse h'01' page 0 ou en h'101' page 2. [3]

I.4.1.1.Mode TIMER

Le choix de ce mode se fait par : TOCS = 0 (b5 de OPTION).

TMR0 est incrémenté à chaque cycle instruction (Fosc/4), en considérant le prédiviseur avec un rapport de 1.

I.4.1.2.Mode COMPTEUR

Ce mode est sélectionné si TOCS = 1. TMR0 est alors Incrémenté à chaque front montant ou descendant sur la broche PA4/CLK (pin3). Le choix du front est fait par le bit TOSE (b4 d'OPTION).

Si TOSE = 0 le compteur s'incrémente à chaque front montant.

Si TOSE = 1 c'est le front descendant qui incrémente le compteur.

I.4.1.3.LE PREDIVISEUR

Il est partagé entre le Watchdog et TMR0.

L'affectation se fait par le bit PSA (b3 d'OPTION).

Si PSA = 0 le prédiviseur est affecté à TMR0.

Le choix du rapport de division se fait avec les bits PS2, PS1 et PS0 (b2, b1 et b0 d'OPTION).

Si PSA = 1 le pré diviseur est affecté au Watchdog et le rapport de division

Pour TMR0 est fixé à 1. [16]

PS2	PS1	PS0	TMR0	WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Tableau I.2. *Taux* pré diviseur. [3]

I.4.1.4.INTERRUPTION DE TMR0

Elle est générée quand TMR0 passe de la valeur FF à 00.

Le Flag TOIF (b2 d'INTCON) passe à "1". On peut masquer la génération de

L'interruption en mettant le bit TOIE (b5 d'INTCON) à "0".

Le Flag TOIF dit être remis à zéro par soft dans le sous-programme

D'interruption, avant de re-autoriser cette interruption.

- Lorsque le bit de PSA est effacé, l'étage d'adaptation est affecté à la Timer0 / compteur, comme illustré sur la figure ci-dessous:

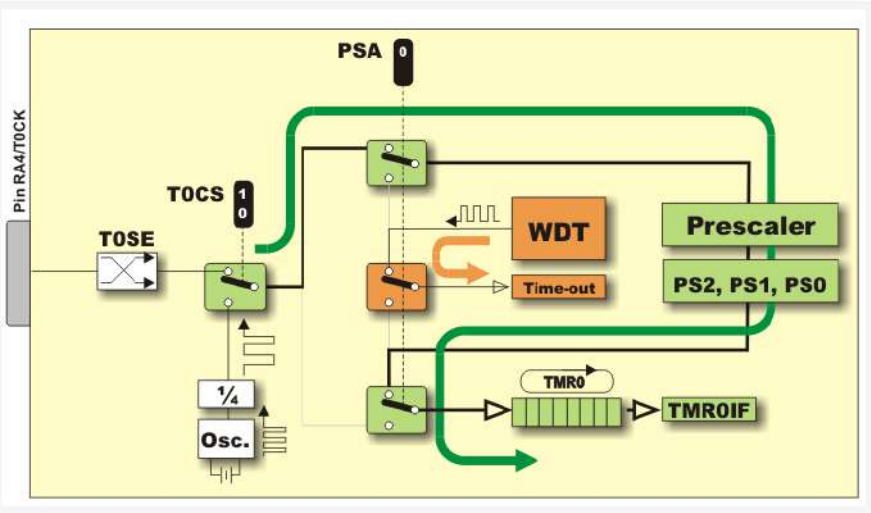


Figure. I-5.*La fonction du PSA bit 0.* [3]

- Lorsque le bit de PSA est réglé, l'étage d'adaptation est affecté à la minuterie de chien de garde comme la montre la figure ci-dessous:

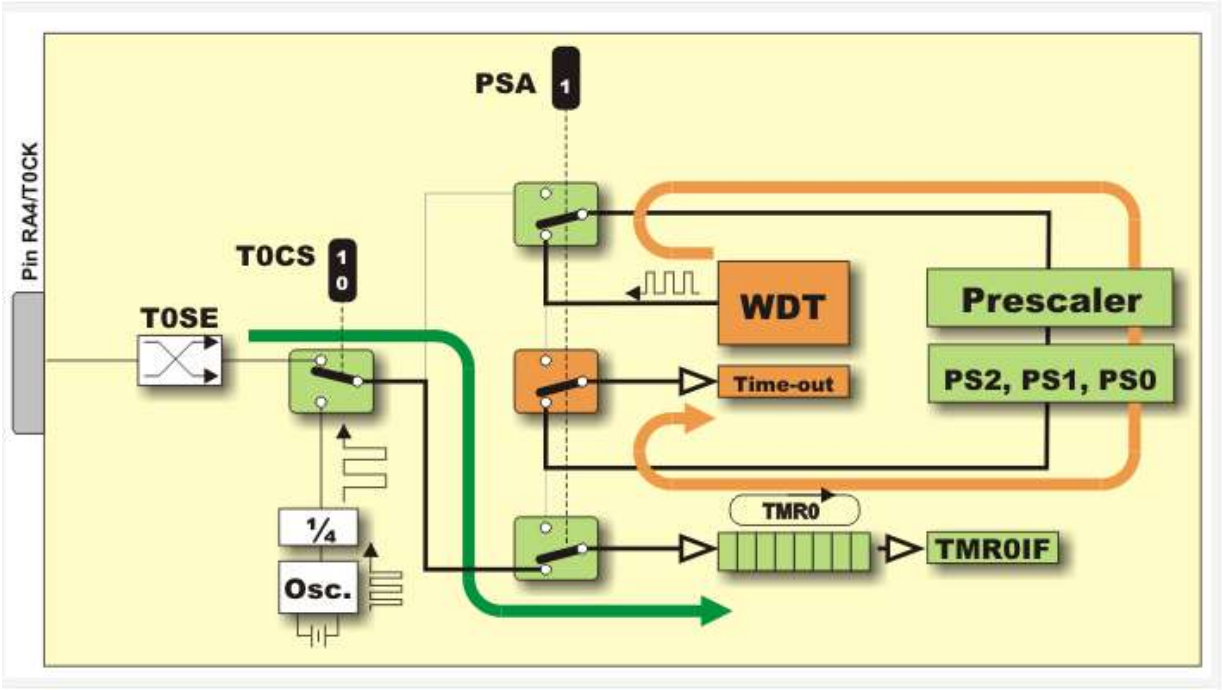


Figure. I.6. *La fonction du PSA bit 1*[3]

a)Cela vaut également mentionner

Lorsque le prédiviseur est affecté à la temporisation / compteur, toute écriture sur le registre TMR0 efface le prédiviseur.

Lorsque le prédiviseur est affecté à la minuterie chien de garde, l'instruction CLRWDT efface à la fois.

Écrivez au registre TMR0, utilisé comme une minuterie, ne causera pas l'impulsion de comptage pour commencer immédiatement, mais avec deux cycles d'instruction retard.

En conséquence, il est ne néces- pour ajuster la valeur écrite dans le registre TMR0. Lorsque le microcontrôleur est configuré en mode veille, l'oscillateur d'horloge est désactivé. Aucun débordement ne peut se produire car il n'y a pas d'impulsions pour compter et donc le débordement d'interruption TMR0 ne peut pas réveiller le microcontrôleur du mode veille.

Lorsqu'il est utilisé comme un compteur d'horloge externe, sans prédiviseur, d'une longueur minimale d'impulsion ou d'un retard entre les deux impulsions doit être tosc 2 + 20ns (Tosc représente la période d'un signal d'horloge généré par l'oscillateur).

Lorsqu'il est utilisé comme un compteur d'horloge externe avec prédiviseur, une longueur minimale d'impulsion ou un intervalle entre deux impulsions est seulement 10ns.

Le registre diviseur 8 bits n'est pas disponible pour l'utilisateur, ce qui signifie qu'il ne peut pas être directement lue ou écrite. [3]

Lors du passage de la cession de prédiviseur de Timer0 à regarder chien minuterie, la séquence d'instructions suivante écrite en langage d'assemblage doit être exécuté afin d'empêcher le microcontrôleur de réinitialisation:

I.4.1.5.Resume

Pour utiliser Timer0 correctement, il est nécessaire:

Etape 1: Pour sélectionner le mode

mode de minuterie est sélectionné par le bit T0CS du registre OPTION_REG, (T0CS: 0 = temporisateur, 1 = compteur).

Lorsqu'il est utilisé, l'étage d'adaptation doit être affectée au temporisateur / compteur en effaçant le bit de PSA du registre à OPTION_REG. Le taux de prédiviseur est réglé en utilisant les bits PS2-ps0 du même registre.

Lorsqu'une interruption est utilisée, les bits et le GIE TMR0IE du registre INTCON devraient être fixés.

Étape 2: Pour mesurer et compter

Mesure du temps:

Réinitialiser le registre TMR0 ou écrire une valeur connue à elle.

Le temps écoulé (en microsecondes si le cristal de quartz 4MHz est utilisé) est mesurée par la lecture du registre TMR0.

Le TMR0IF bit drapeau du registre INTCON est automatiquement réglé chaque fois que le

registre TMR0 débordements. Si elle est activée, une interruption se produit.

Compte:

La polarité des impulsions à compter sur la broche RA4 est sélectionné par le bit TOSE du registre OPTION_REG (T0SE: 0 = positif, 1 = impulsions négatives). Nombre d'impulsions peut être lue à partir du registre TMR0. L'étage d'adaptation et d'interruption sont utilisés de la même manière que dans le mode de temporisateur. [3]

I .4.2.timer1:

Module timer1 est un 16-bit temporisateur / compteur, ce qui signifie qu'il se compose de deux registres (TMR1L et TMR1H). Il est capable de compter jusqu'à 65.535 impulsions dans un cycle unique, à savoir avant que le comptage commence à partir de zéro

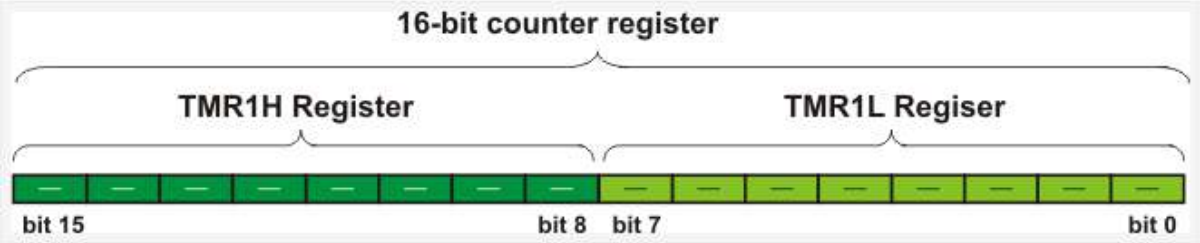


Figure. I.7. *minuteur timer1* [16]

Semblable à Timer0, ces registres peuvent être lus ou écrits en tout point. En cas de débordement, une interruption est générée automatiquement, à condition qu'il soit activé.

Le Minuteur1 peut être réglé pour fonctionner dans l'un des deux modes, soit sous forme d'une minuterie ou un compteur. A la différence du Timer0, ces deux modes donnent possibilités supplémentaires. [3]

I .4.2.1. Présente les caractéristiques de Timer1 suivantes

- 16-bit paire temporisateur / compteur de registre;
- la source d'horloge interne ou externe programmable;
-pré diviseur 3 bits;
-Low Power (LP) oscillateur en option;
- fonctionnement synchrone ou asynchrone;
-Timer1 commande de grille (COUNT activée) via comparateur ou la broche T1G;
-Interrompre en cas de débordement;

-Wake-up on overflow (horloge externe); et

- Source d'horloge pour Capture / Comparer module.

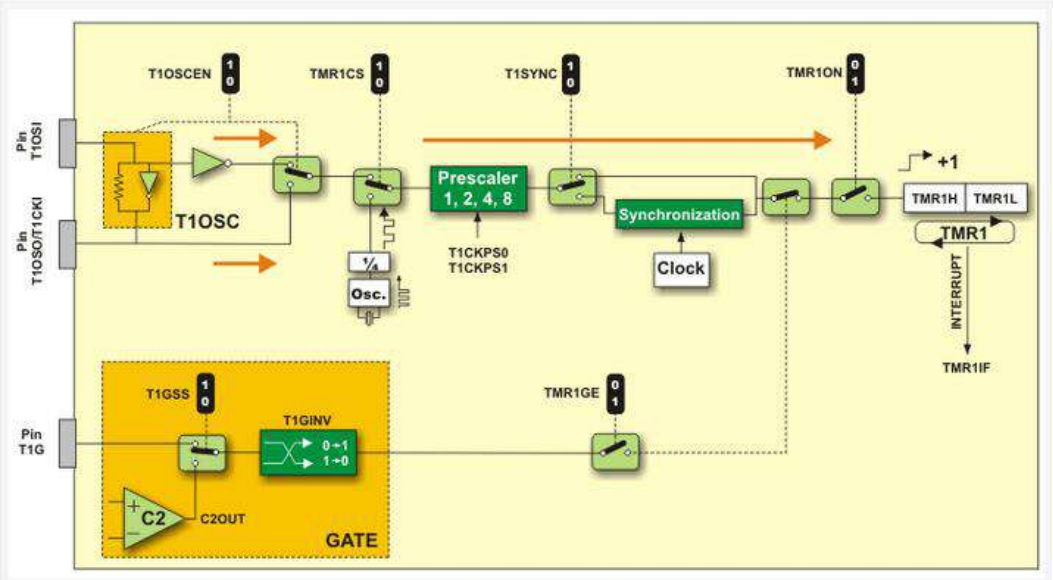


Figure. I.8. *minuteur timer1 vue d'ensemble. [3]*

I .4.2.2. Minuteur TMR1 pré diviseur

Minuteur TMR1 a un pré diviseur complètement séparé qui permet à 1, 2, 4 ou 8 divisions de l'entrée d'horloge. Le pré diviseur n'est pas directement lisible ou inscriptible. Cependant, le compteur diviseur est automatiquement effacé à écrire au registre TMR1H ou TMR1L.

I .4.2.3. Minuteur TMR1 Oscillateur

RC0 / T1OSO et RC1 / T1OSI broches sont utilisés pour enregistrer des impulsions provenant de l'électronique périphériques, mais ils ont aussi une fonction supplémentaire. Comme on le voit dans la figure 4-7, ils sont simultanément configurés à la fois comme entrée (broche RC1) et de sortie (broche RC0) de l'oscillateur à quartz LP supplémentaire (de faible puissance).

Ce circuit additionnel est principalement conçu pour fonctionner à des fréquences basses (jusqu'à 200 KHz), et plus précisément, pour l'utilisation du cristal de quartz 32,768 KHz. Ces cristaux sont utilisés dans les montres à quartz car il est facile d'obtenir une deuxième impulsion longue en divisant simplement cette fréquence.

Depuis cet oscillateur ne dépend pas de synchronisation interne, il peut fonctionner même

en mode veille. Il est activé en réglant le bit de contrôle T1OSCEN du registre T1CON. L'utilisateur doit fournir un retard de temps de logiciel (quelques millisecondes) pour assurer un bon oscillateur de démarrage. [3]

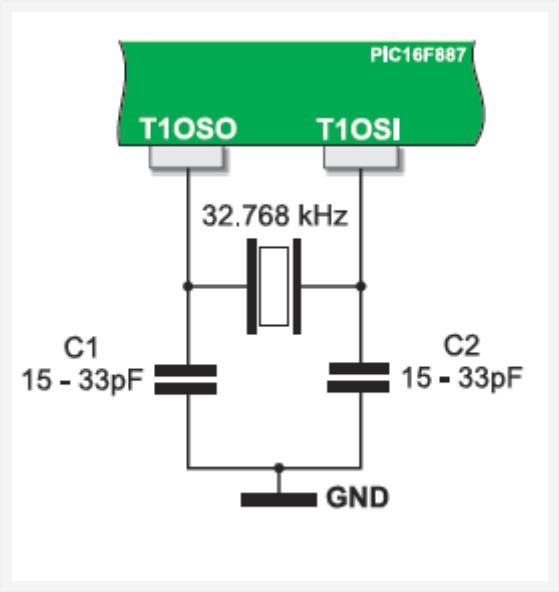


Figure .I.9. *TMR1 Oscillateur. [3]*

Le tableau ci-dessous montre les valeurs recommandées de condensateurs en fonction de l'oscillateur à quartz. Ces valeurs ne doivent pas être exactes. Toutefois, la règle générale est la suivante: la capacité du condensateur élevée plus la stabilité, ce qui, en même temps, de prolonger le temps nécessaire pour la stabilité de l'oscillateur.

OSCILLATOR	FREQUENCY	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF

Tableau I.3. *La valeur recommandée de condensateurs en fonction de l'oscillateur à quartz.*

I .4.2.4. TMR1 Porte

Minuteur 1 grille-source est un logiciel configurable pour être la broche T1G ou la sortie du comparateur C2. Cette porte permet la minuterie au temps directement des événements externes à l'aide de l'état logique sur la broche T1G ou des événements analogiques en utilisant la sortie

du comparateur C2. Voir la figure 4-7 ci-dessus. Pour en temps, une durée de signaux, il est suffisant pour permettre à cette porte et compter les impulsions ayant passé à travers elle.

I .4.2.5. TMR1 en mode de minuterie

Pour sélectionner ce mode, il est nécessaire d'effacer le bit TMR1CS. Après cela, le registre à 16 bits est incrémenté à chaque impulsion provenant de l'oscillateur interne. Si le cristal de quartz 4MHz est utilisé, il sera incrémenté à chaque microseconde.

Dans ce mode, le bit T1SYNC n'a aucune incidence sur la minuterie, car il compte les impulsions d'horloge interne. Etant donné que l'ensemble de l'électronique utilise ces impulsions, il n'y a pas besoin de la synchronisation. [3]

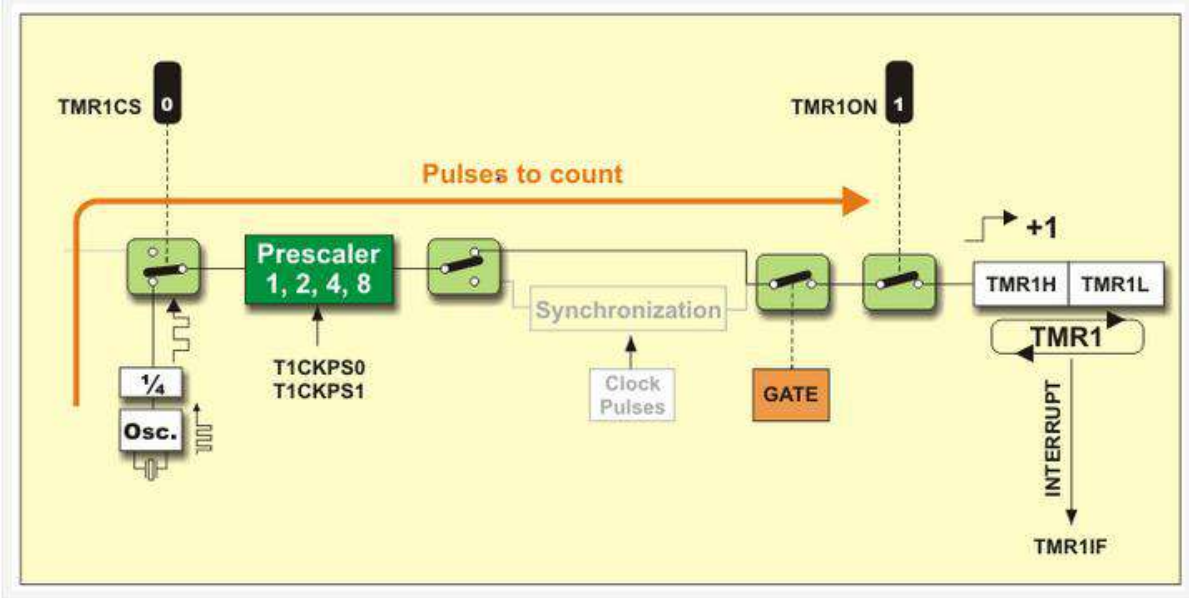


Figure .I.10.*TMR1 en mode de minuterie. [3]*

Horloge de l'oscillateur du microcontrôleur ne fonctionne pas en mode veille de sorte que le débordement de registre de minuterie ne peut causer aucune interruption.

I .4.2.6. Minuteur TMR1 Oscillateur

La consommation électrique du microcontrôleur est réduit au niveau le plus bas en mode veille. Le point est d'arrêter l'oscillateur. Quoi qu'il en soit, il est facile de régler la minuterie dans ce mode- en écrivant une instruction SLEEP pour le programme. Un problème se produit quand il est nécessaire de se réveiller le microcontrôleur car seule une interruption peut le faire. Depuis le microcontrôleur "dort", une interruption doit être déclenchée par l'électronique

externe. Tout cela peut se incroyablement compliqué si elle est nécessaire, le «réveil» se produit à des intervalles de temps réguliers.

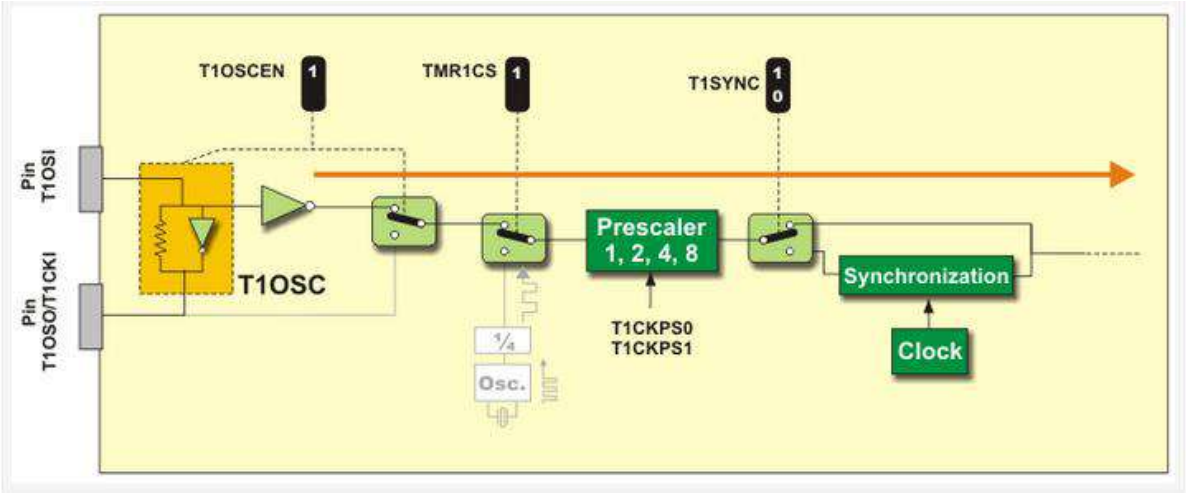


Figure .I.11. *TMR1 Oscillateur. [3]*

Afin de résoudre ce problème, un oscillateur bas puissance de quartz complètement indépendant, capable de fonctionner en mode veille, est intégré dans le microcontrôleur PIC16F887. Simplement, ce qui auparavant a été un circuit séparé, il est maintenant intégré dans le microcontrôleur et affecté à la minuterie TMR1. L'oscillateur est activé en réglant le bit T1OSCEN du registre T1CON. Après cela, le bit TMR1CS du même registre, puis est utilisée pour déterminer que la minuterie TMR1 utilise des séquences d'impulsions à partir de cet oscillateur.

Le signal issu de cet oscillateur à quartz est synchronisé avec l'horloge du microcontrôleur en effaçant le bit T1SYNC. Dans ce cas, la minuterie ne peut pas fonctionner en mode veille. Vous vous demandez pourquoi? Parce que le circuit de synchronisation utilise l'horloge du microcontrôleur !et

- Le débordement de registre d'interruption TMR1 peut être activé. Ces interruptions auront lieu en mode veille ainsi. [3]

I .4.2. 7. TIMER1 en mode compteur

TMR1 minuterie commence à fonctionner en tant que compteur en positionnant le bit TMR1CS. Cela signifie que le temporisateur est incrémenté TMR1 sur le front montant de l'T1CKI d'entrée d'horloge externe. Si le bit de contrôle T1SYNC du registre T1CON est effacé,

les entrées d'horloge externes seront synchronisées sur leur chemin vers le registre TMR1. En d'autres termes, le temporisateur TMR1 est synchronisé sur l'horloge système du microcontrôleur et appelé compteur synchrone.

Lorsque le microcontrôleur, opérant de cette manière, est réglé en mode veille, les registres temporisateurs TMR1H et TMR1L ne sont pas incrémenté même si des impulsions d'horloge apparaissent sur les broches d'entrée. Simplement, étant donné que l'horloge du système à microcontrôleur ne fonctionne pas dans ce mode, il n'y a pas d'entrées d'horloge à utiliser pour la synchronisation. Cependant, le prédiviseur continuera à fonctionner s'il y a des impulsions d'horloge sur les broches car il est juste un diviseur de fréquence simple. [3]

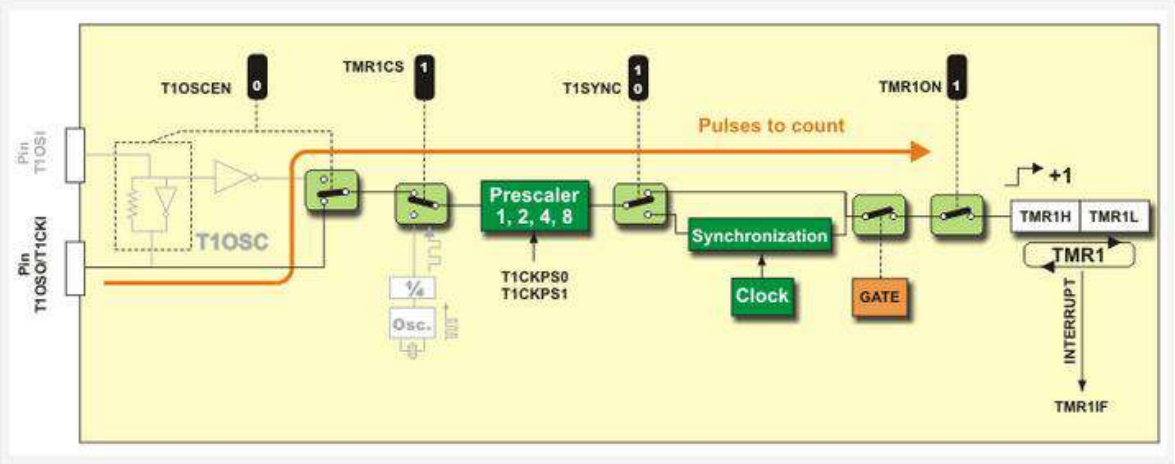
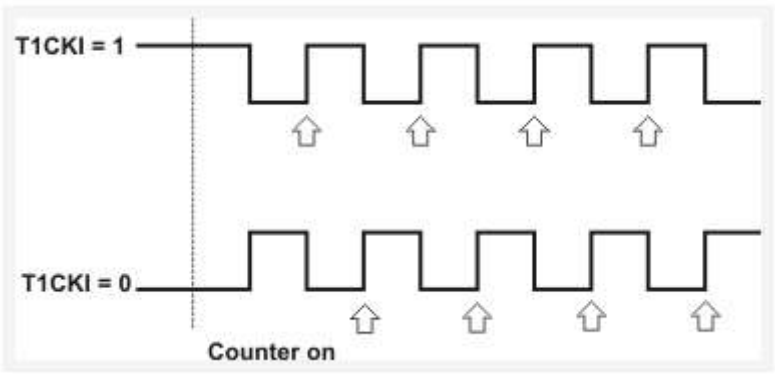


Figure .I.12. *Mode compteur. [3]*

Ce compteur enregistre une logique (1) sur les broches d'entrée. Il est important de comprendre qu'au moins un front descendant doit être enregistré avant la première augmentation sur le front montant. Voir la figure à gauche. Les flèches de la figure 4-11 indiquent incrémente.



T1CON Register



Figure .I.13. *T1CON Register. [3]*

T1GINV - Timer1 Porte Invert bit agit comme état logique inverseur sur la grille de broche T1G ou la porte sortie du comparateur C2 (C2OUT). Elle permet à la minuterie pour assurer Mea temps alors que la porte est haute ou basse.

1 - Minuteur 1 compte lorsque la broche T1G ou porte bit C2OUT est élevé (1); et
0 - Minuteur 1 compte lorsque la porte de la broche T1G ou bit C2OUT est faible (0).

TMR1GE - Timer1 Porte Activer bit détermine si la porte de sortie broche T1G ou comparateur C2 (C2OUT) sera actif ou non. Ce bit est fonctionnelle uniquement dans le cas où la minuterie TMR1 est activé (bit TMR1ON = 1). Sinon, ce bit est ignoré.

1 Minuteur TMR1 est que si la minuterie 1 porte est non actif; et
0 Porte n'a aucune incidence sur la minuterie TMR1.

T1CKPS1, T1CKPS0 - Timer1 entrée Horloge Prescale Sélectionnez les bits de déterminer le taux du prédiviseur affecté à la minuterie TMR1.

T1CKPS1	T1CKPS0	PRESCALER RATE
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

Tableau I.4. Prescaler Taux de prediviseur. [3]

T1OSCEN - LP Oscillateur Activer bit de contrôle

1 - LP oscillateur est activé pour minuterie TMR1 horloge (oscillateur à faible consommation d'énergie et la fréquence de 32,768 kHz); et

0 - LP oscillateur est éteint.

T1SYNC - Timer1 horloge externe Entrée bit de contrôle de synchronisation permet la synchronisation de l'entrée de l'oscillateur LP ou T1CKI entrée de la broche avec l'horloge interne du microcontrôleur. Lors du comptage des impulsions provenant de la source d'horloge locale (bit TMR1CS = 0), ce bit est ignoré.

1 - Ne pas synchroniser l'entrée d'horloge externe; et

0 - Synchroniser entrée d'horloge externe.

TMR1CS - Minuteur TMR1 Clock Source Select bit

1 - compte des impulsions sur la broche T1CKI (sur le front montant 0-1); et

0 - compte des impulsions de l'horloge interne du microcontrôleur.

TMR1ON - Timer1 Le bit

1 - Active Minuteur TMR1; et

0 - Arrête la minuterie TMR1.

Pour utiliser la minuterie TMR1 correctement, il est nécessaire d'effectuer les opérations suivantes:

Comme il est impossible de désactiver le prédiviseur, son taux doit être ajustée en utilisant les bits T1CKPS1 et T1CKPS0 du T1CON de registre (Voir le tableau 4-2);

Le mode doit être sélectionné par le bit TMR1CS du même registre (TMR1CS: 0 = la source d'horloge est un oscillateur à quartz, 1 = la source d'horloge est fournie à l'extérieur);

En réglant le bit T1OSCEN du même registre, la minuterie TMR1 est sous tension et les

registres TMR1H et TMR1L sont incrémenté à chaque entrée d'horloge. Compter arrêts en désactivant ce bit;

Le prédiviseur est effacé en effaçant ou en écrivant les registres de compteur; et En remplissant les deux registres de minuterie, le TMR1IF drapeau est réglé et le comptage commence à partir de zéro. [3]

L'équation qui permet de calcule T est donner comme suite :

$T = 4 * T_h *$ pré division $*(65535\text{-}VI)$ **(III.3)**

Ou : T_h : période d'horloge

VI : valeur initiale

On a une fréquence de 8MHz ⟶ donc $T_h = 1/8*10^6$

I .4.2.8. Résumé :

Pour utiliser le Timer1 correctement, il est nécessaire d'effectuer les opérations suivantes:

Comme il est impossible de désactiver le prédiviseur, son taux doit être ajustée en utilisant les bits T1CKPS1 et T1CKPS0 du T1CON de registre (Reportez-vous au tableau précédent).

Sélectionnez le mode approprié en utilisant le bit TMR1CS du T1CON de registre. (TMR1CS: 0 = la source d'horloge est un oscillateur à quartz interne, 1 = l'horloge est fournie de l'extérieur).

En réglant le bit T1OSCEN du même registre, l'oscillateur est activé et que les registres TMR1H et TMR1L sont incrémenté à chaque impulsion d'horloge. Compter arrêts en désactivant ce bit.

Le prédiviseur est effacé en effaçant ou en écrivant aux registres de compteur.

Lorsque les deux registres temporisateurs sont remplis avec ceux, un débordement a lieu, le drapeau TMR1IF est réglé et le comptage commence à partir de zéro. [3]

I .5. Les interruptions

Quand une demande d'interruption arrive, il ne signifie pas que d'interruption se fera automatiquement, car il doit aussi être activée par l'utilisateur (à partir du programme).

A cause de cela, il y a des bits spéciaux utilisés pour activer ou désactiver les interruptions.

Il est facile de reconnaître ces bits par IE contenus dans leurs noms (signifie Interrupt Enable). Par ailleurs, chaque interruption est associé à un autre bit appelé drapeau qui indique que la demande d'interruption est arrivée, peu importe que ce soit activé ou non.

Ils sont aussi facilement reconnaissables par les deux dernières lettres contenues dans leur names- IF (Interrupt Flag).

Comme on le voit, tout est basé sur une idée simple et efficace. Quand une demande d'interruption arrive, le bit de drapeau est d'abord être réglé.

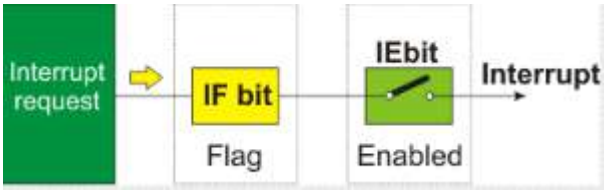


Figure. I.14. *Interrupt Registres système*. [4]

Si le bit IE approprié est pas défini (0), cet événement sera complètement ignoré. Sinon, une interruption se produit! Au cas où plusieurs sources d'interruption sont activées, il est nécessaire de détecter une activité avant le début de la routine d'interruption exécution.

La détection de la source est effectuée en vérifiant les bits de drapeaux.

Il est important de comprendre que les bits de drapeaux ne sont pas effacés automatiquement, mais par le logiciel lors de la routine d'interruption exécution. Si ce détail est négligé, une autre interruption aura lieu immédiatement après le retour au programme, même si il n'y a pas plus de demandes pour son exécution! Autrement dit, le drapeau ainsi que IE bit sont restés fixés.

Toutes les sources d'interruptions typiques du microcontrôleur PIC16F887 sont indiquées sur la page suivante. Remarque plusieurs choses:

bit GIE- Permet toutes les interruptions non masquées et désactive toutes les interruptions simultanément. [4]

PEIE bit - Permet toutes les interruptions périphériques non masquées et désactive toutes les interruptions périphériques (Ceci ne concerne pas Minuteur TMR0 et le port B interrompent les sources).

Pour activer interruption causée par changement d'état logique sur le port B, il est nécessaire de l'activer pour chaque bit séparément. Dans ce cas, les bits de laIOCBregistre ont la fonction de contrôler les bits d'IE

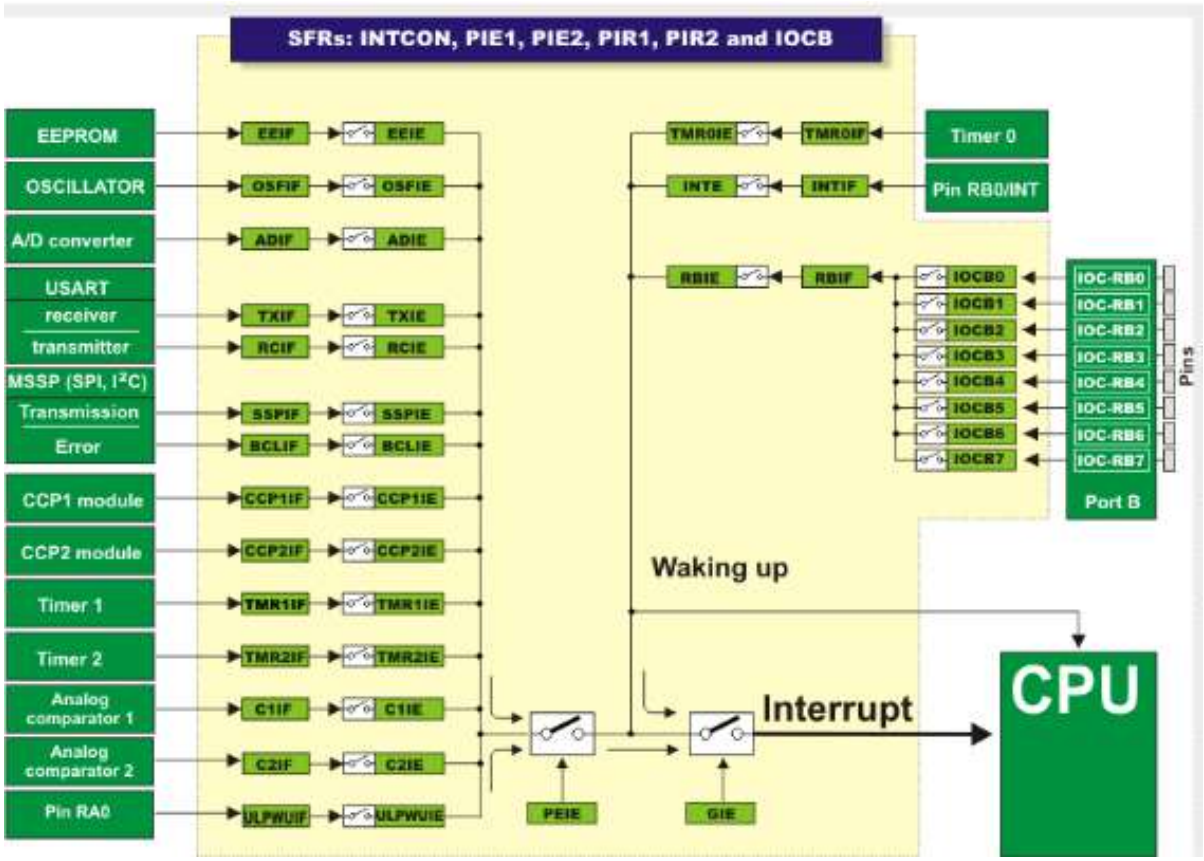


Figure. I.15. *Interrompre SFRs* [4]

INTCON Register

Le registre INTCON contient diverses activer et drapeau bits pour TMR0 débordement de registre, PORTB changement et les interruptions de broches INT externes.

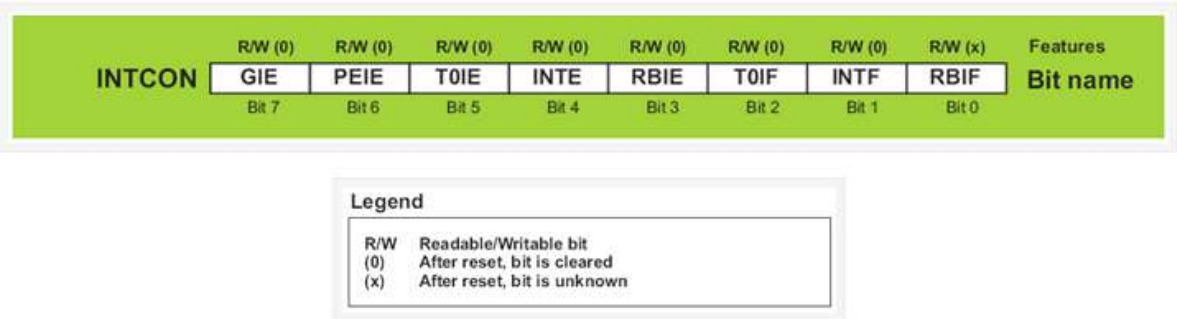


Figure. I.16. *INTCON Register. [4]*

GIE - Global Interrupt Enable bit - contrôle toutes les sources d'interruption possibles simultanément.

1 - Active toutes les interruptions non masquées.

0 - Désactive toutes les interruptions.

PEIE - Peripheral Interrupt Enable bit agit semblable au GIE, mais les contrôles interruptions activé par les périphériques. Cela signifie qu'il n'a pas d'incidence les interruptions déclenchées par la minuterie TMR0 ou en changeant d'état sur le port B ou broche RB0 / INT.

1 - Active toutes les interruptions périphériques non masquées.

0 - Désactive toutes les interruptions périphériques.

T0IE - TMR0 Overflow Interrompre Activer les contrôles de bit d'interruption activés par TMR0 débordement.

1 - Active l'interruption TMR0.

0 - Désactive l'interruption TMR0.

INTE - RB0 / INT Interruption externe Activer contrôle de bit d'interruption causés par changement d'état logique sur la broche RB0 / IN (interruption externe).

1 - Active l'interruption externe INT.

0 - Désactive l'interruption externe INT.

Rbie - RB Port Change Interrompre Activer bit. Lorsqu'il est configuré en entrée, le port broches B peuvent provoquer des interruptions en changeant leur état logique (peu importe si elle est faible transition highto- ou vice versa, fait que quelque chose est changé seulement les questions). Ce bit détermine si interruption doit avoir lieu ou non.

1 - Active le port B change d'interruption.

0 - Désactive le port B changent d'interruption.

T0IF - TMR0 Overflow Flag Interrompre bits enregistre le minuteur TMR0 registre de débordement, lors du comptage commence à partir de zéro.

1 - TMR0 registre a débordé (bit doit être effacé dans le logiciel).

0 - TMR0 registre n'a pas débordé.

INTF - RB0 / INT Flag Interrompre externe bit registres changement d'état logique sur la broche RB0 / INT.

1 - L'interruption externe INT a eu lieu (doit être effacé dans le logiciel).

0 - L'interruption externe INT n'a pas eu lieu.

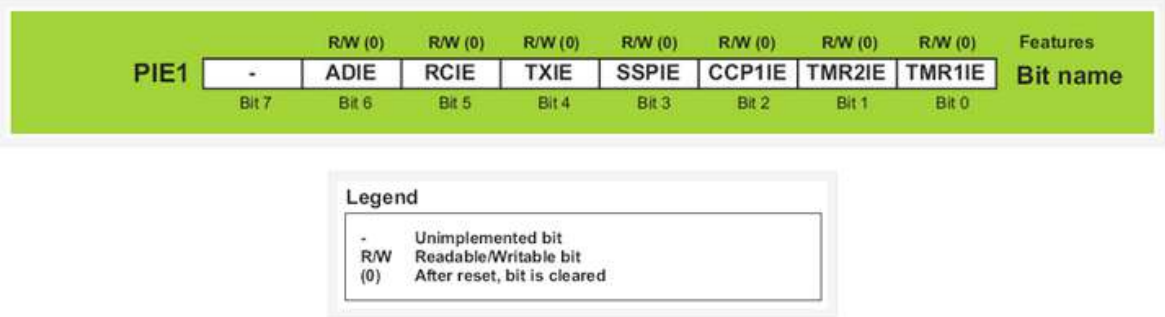
RBIF - RB Port Change Interrupt Flag bit enregistre un changement d'état de certaines broches d'entrée Port B logique.

1 - Au moins un port B usage général I / O broches a changé d'état. À la lecture de PortB, RBIF (de bit drapeau) doit être effacé dans le logiciel.

0 - None du port B usage général I / O broches a changé d'état. [4]

PIE1 Register

Le registre PIE1 contient le périphérique d'interruption bits de validation.



Figue. I.17. registre PIE1. [4]

ADIE - Convertisseur A / N Interruption Activer bit.
1 - Active l'interruption de l'ADC.
0 - Désactive l'interruption ADC.

RCIÉ - EUSART Recevoir Interrompre Activer bit.
1 - Active le EUSART recevoir interruption.
0 - Désactive la EUSART recevoir interruption.

TXIE - EUSART Transmit Interrupt Enable bit.
1 - Active l'interruption EUSART d'émission.
0 - Désactive l'interruption EUSART d'émission.

SSPIE - Master Synchronous Serial Port (MSSP) Interrompre Activer bit - permet à une demande d'interruption doit être généré après chaque transfert de données via série synchrone module de communication (ou en mode I2C SPI).
1 - Active l'interruption MSSP.
0 - Désactive l'interruption MSSP.

CCP1IE - Interruption CCP1 bit d'activation permet à une demande d'interruption doit être générée dans le module de CCP1 utilisé pour le traitement du signal PWM.
1 - Active l'interruption de CCP1.
0 - Désactive l'interruption CCP1.

TMR2IE - TMR2 à PR2 match Interrompre Activer bit
1 - Active le match d'interruption TMR2 à PR2.
0 - Désactive le match d'interruption TMR2 à PR2.

TMR1IE - TMR1 Overflow Interrompre Activer bit permet une demande d'interruption doit être généré après chaque temporisateur TMR1 registre de débordement, à savoir lorsque le comptage commence à partir de zéro.
1 - Active le trop-plein d'interruption TMR1.
0 - Désactive le trop-plein d'interruption TMR1. [4]

PIE2 Register

Le PIE2 registre contient également les différents bits de validation d'interruption.

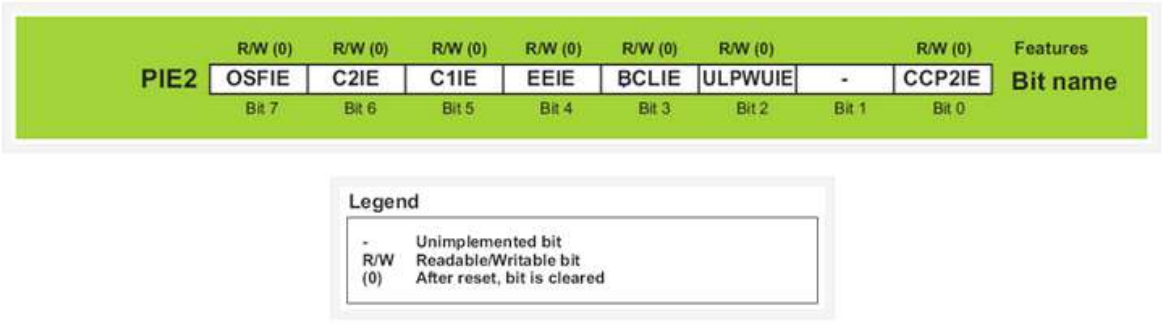


Figure. I.18. PIE2 Register. [4]

OSFIE - Oscillateur Fail Interrompre Activer bit.
1 - Active oscillateur échec d'interruption.
0 - Désactive oscillateur échec d'interruption.

C2IE - Comparator C2 Interrupt Enable bit.
1 - Active Comparator C2 interruption.
0 - Désactive Comparator C2 interruption.

C1IE - Comparator C1 Interrupt Enable bit.
1 - Active Comparator C1 interruption.
0 - Désactive Comparator C1 interruption.

EEIE - EEPROM Ecrire Opération Interrompre Activer bit.
1 - Active le fonctionnement EEPROM écriture d'interruption.
0 - Désactive opération EEPROM écriture d'interruption.

BCLIE - Bus Collision Interrompre Activer bit.
1 - Active bus collision interruption.
0 - Désactive bus collision interruption.

ULPWUIE - Ultra Low-Power Wake-up Interrompre Activer bit.
1 - Active Low-Power Wake-up interruption Ultra.
0 - Désactive Ultra Low-Power Wake-up d'interruption.

CCP2IE - CCP2 Interrompre Activer bit.
1 - Active CCP2 interruption.
0 - Désactive CCP2 interruption. [4]

PIR1 Register

Le registre PIR1 contient les bits de drapeau d'interruption.

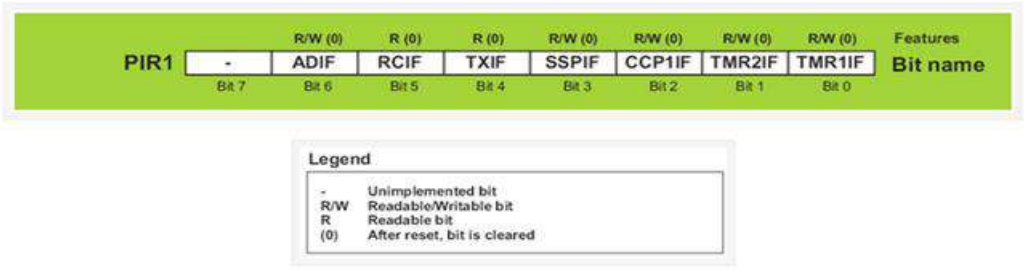


Figure. I.19.PIR1 Register. [4]

ADIF - A bit / D Converter Interrupt Flag.
1 - conversion A / D est terminée (bit doit être effacé dans le logiciel).
0 - conversion A / D est pas remplie ou n'a pas commencé.

RCIF - EUSART Recevez Interrupt Flag bit.
1 - Le EUSART tampon de réception est pleine. Bit est effacé par la lecture du registre RCREG.
0 - Le EUSART tampon de réception n'est pas plein.

TXIF - EUSART Transmit Interrupt Flag bit.
1 - Le tampon EUSART d'émission est vide. Bit est effacé par écrit au registre TXREG.
0 - Le tampon EUSART d'émission est plein.

SSPIF - Port série Maître synchrone bit (MSSP) Interrupt Flag.
1 - Le MSSP d'interruption condition pendant les données d'émission / réception a eu lieu. Ces conditions varient selon le mode de fonctionnement MSSP (SPI ou I2C) Ce bit doit être effacé dans le logiciel avant de retourner à la routine de service d'interruption.
0 - Aucune condition MSSP d'interruption n'a eu lieu.

CCP1IF - CCP1 Interrupt Flag bits.
1 - CCP1 condition d'interruption est survenue (CCP1 est l'unité de capture, de comparaison et de génération de signal PWM). Selon le mode de fonctionnement, la capture ou comparer match a eu lieu. Dans les deux cas, le bit doit être effacé dans le logiciel. Ce bit n'est pas utilisé dans le mode PWM.
0 - Aucune condition CCP1 d'interruption n'a eu lieu.

TMR2IF - Timer2 à PR2 Interrupt Flag bit
1 - TMR2 (registre de 8 bits) pour PR2 correspondance est survenue. Ce bit doit être effacé dans le logiciel avant de retourner à la routine de service d'interruption.
0 - Non TMR2 à PR2 rencontre a eu lieu.

TMR1IF - Timer1 Overflow Flag Interrompre bit
1 - Le registre TMR1 a débordé. Ce bit doit être effacé dans le logiciel.
0 - Le registre TMR1 n'a pas débordé. [4]

PIR2 Register

Le registre PIR2 contient les bits de drapeau d'interruption.



Figure. I.20. registre PIR2. [4]

OSFIF - Oscillateur Fail Interrupt Flag bit.

1 - oscillateur système des Nations Unies et Échoué d'horloge d'entrée non changement d'Écoulement INTOSC oscillateur interne. Bit Ce Doit Être effacé dans le logiciel.

0 - oscillateur système Fonctionne normalement.

C2IF - Bits Comparator C2 Flag Interrompre.

1 - la sortie C2 du comparateur non changement (bits de C2OUT). Bit Ce Doit Être effacé dans le logiciel.

0 - pas sortie changeur C2 Comparator de N'a.

C1IF - Comparator C1 Interrupt Flag bit.

1 - la sortie C1 du comparateur non changement (bits de C1OUT). Bit Ce Doit Être effacé dans le logiciel.

0 - pas sortie changeur C1 Comparator de N'a.

EEIF - EE Ecrire Opération Interrupt Flag bit.

1 - EEPROM Terminée écriture. Bit Ce Doit Être effacé dans le logiciel.

0 - EEPROM est écriture pas remplié ous pas de n'à Commencer.

BCLIF - Bus Collision Interrupt Flag bit.

1 - Une collision de bus un lieu eu dans le MSSP est lorsqu'il CONFIGURE le mode de I2C versez maître. Bit Ce Doit Être effacé dans le logiciel.

0 - Pas de bus collision non-lieu eu.

ULPWUIF - Ultra Low-power Wake-up Interrupt Flag bit.

1 - condition de réveil non-lieu eu. Bit Ce Doit Être effacé dans le logiciel.

0 - Aucune état de réveil lieu non eu.

CCP2IF - bits de CCP2 Flag Interrompre.

1 - condition de CCP2 d'interruption a eu lieu (unité de capture, de comparaison et de génération de Signaux PWM). SELON mode le de fonctionnement, la capture ous comparateur correspondre à la place non eu. Dans les Deux CAS, le bit Doït Être effacé dans le logiciel. Bit C'est utilisé dans le pas en mode de PWM.

0 - Aucune état CCP2 d'interruption lieu non eu. [4]

Enregistrer PCON

Le registre de PCON ne contient Que Deux Bits de drapeau Verser utilisé Entre juin

différencier: mise sous tension de réinitialisation, reset brun-out, réinitialiser Watchdog Timer et externe (la part broche MCLR).



Figure.I.21. registre 2-15 PCON. [4]

ULPWUE - Ultra Low-Power Wake-up Activer bit
1 - Ultra Low-Power Wake-up activé.
0 - Ultra Low-Power réveil désactivé.

SBOREN - Logiciel BOR Activer bit
1 - Brown-out Réinitialiser activé.
0 - Brown-out Réinitialiser désactivé.

POR - Réinitialiser Bit d'état de mise sous tension
1 - Pas d'alimentation de réinitialisation a eu lieu.
0 - Power-on reset a eu lieu. Ce bit doit être réglé dans le logiciel après une réinitialisation de mise sous tension se produit.

BOR - Brown-out bit d'état Réinitialiser
1 - Pas de Brown-out reset a eu lieu.
0 - Brown-out reset a eu lieu. Ce bit doit être réglé dans le logiciel après une réinitialisation Brown se produit. [4]

TMR0I : Dépassement du TIMER0 « débordement le passage de **255** à **0** »

TMR1I : Dépassement du TIMER1 « débordement le passage de **65535** à **0** »

TMR2I : Contenu du PR2 du TIMER2 « débordement le passage de la valeur qui contient dans le registre **PR2** à **0** »

INT : Interruption extérieure RB0

RBI : Changement d'état du port B « RB7 : RB4 »

PSPI : Lecture / Ecriture sur le port parallèle

ADI : Convertisseur Analogique / Numérique

TXI : Emission sur le port RS232

RCI : Réception sur le port RS232

SSPI: Bus SPI / I²C

CCP1I: Capture / Compare / PWM

CCP2I : Capture / Compare / PWM

EEI : Ecriture dans EEPROM

CMI : Comparateur

BCLI : Collision sur Bus [6]

I.5.1. Les ports d'entré sorties

Le microcontrôleur **16F877A** dispose **5 PORT « A, B, C, D, E ».** Tous ces ports sont bidirectionnels.

La plupart des lignes de **Ports** ont une double fonction.
Le **PORT A « 5 bits » I/O** pure et/ou convertisseur analogique et/ou **TIMER0.**
La broche **RA4** du **PORT A** « Entrée du **TIMER T0CKI »** est du type **DRAIN OUVERT**

Le **PORT B « 8 bits » I/O** pure et/ou programmation **ICSP/ICD** « broche **RB3/PGM, RB6/PGC et RB7/PGD** » et l'entrée d'interruption externe **RB0/INT.**

Le **PORT C « 8 bits » I/O** :C'est un port tout ce qu'il a de plus classique, or qu'il a Deux pins qu'on utilisera plus tard dans la communication série avec le PC à travers **« TX** et **RX ».**

Le **PORT D « 8 bits » I/O** Le registre **TRISD** comportera donc les 8bits de direction, pendant que le registre port D correspond aux pins I/O concernés. D'où les 8 pins I/O, en mode entrée, sont du type « trigger de Schmitt ».

Le **PORT E « 3 bits » I/O** Ce port n'est présent que sur les PIC 16F877. Il ne comporte que 3 pins **RE0** à **RE2**, mais, contrairement aux ports, les bits non concernés de TRISE
Sont, cette fois, implantées pour d'autres fonctions. Les pins REX peuvent également être utilisés comme pins d'entrées analogiques. D'où le registre ADCON1 qui détermine si ce port est utilisé comme port I/O ou comme port analogique. [1]

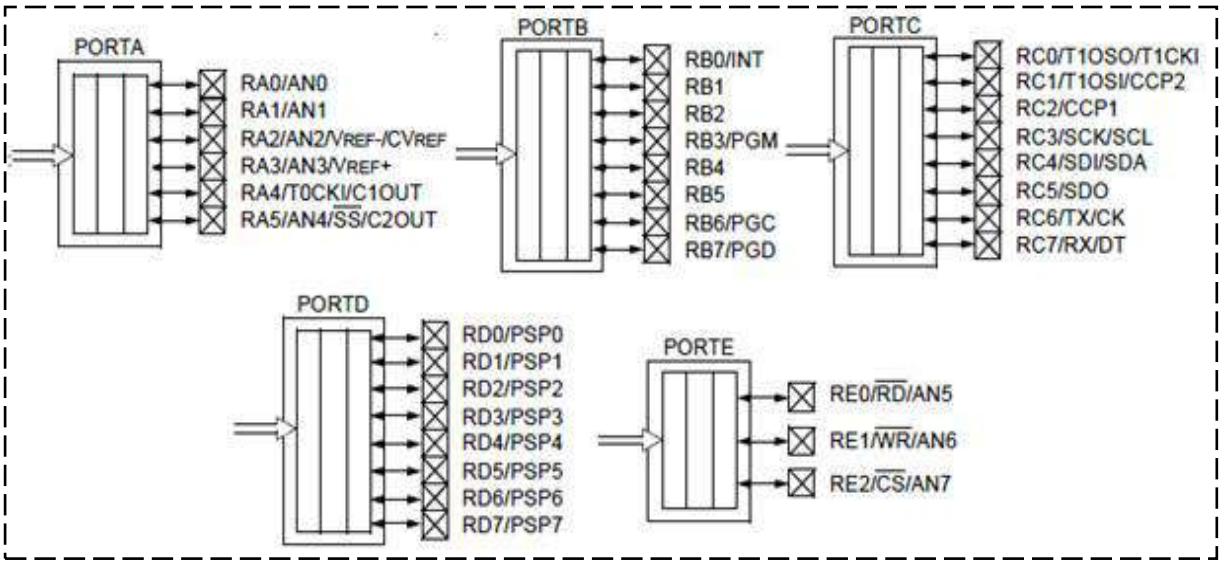


Figure I.22. *Les différents PORT de PIC 16F877A.* [1]

I.5.2. Configuration des portx, les registres PORTX et TRISX

Tous les ports sont pilotés par deux registres :
Le registre **PORTx,** si le **PORTx** ou certaines lignes de **PORTx** sont configurés en sortie, ce registre détermine l'état logique des sorties.

Le registre **TRISx**, c'est le registre de direction. Il détermine si le PORTx ou certaines lignes de port sont en entrée ou en sortie. L'écriture d'un 1 logique correspond à une **entrée** « **1** comme **I**nput » et l'écriture d'un **0** logique correspond à une sortie « **0** comme **O**utput ».Au **RESET** toutes les lignes de ports sont configures en sorties. [1]

I.5.3. Le convertisseur analogique numérique

Notre **16F877A** travaille avec un convertisseur analogique/numérique qui permet un échantillonnage sur 10 bits. Il est composé de :

- Un multiplexeur analogique **5** voies **« PIC 16F877A ».**
- Un échantillonneur bloqueur.
- Un convertisseur analogique numérique de **10** bits.

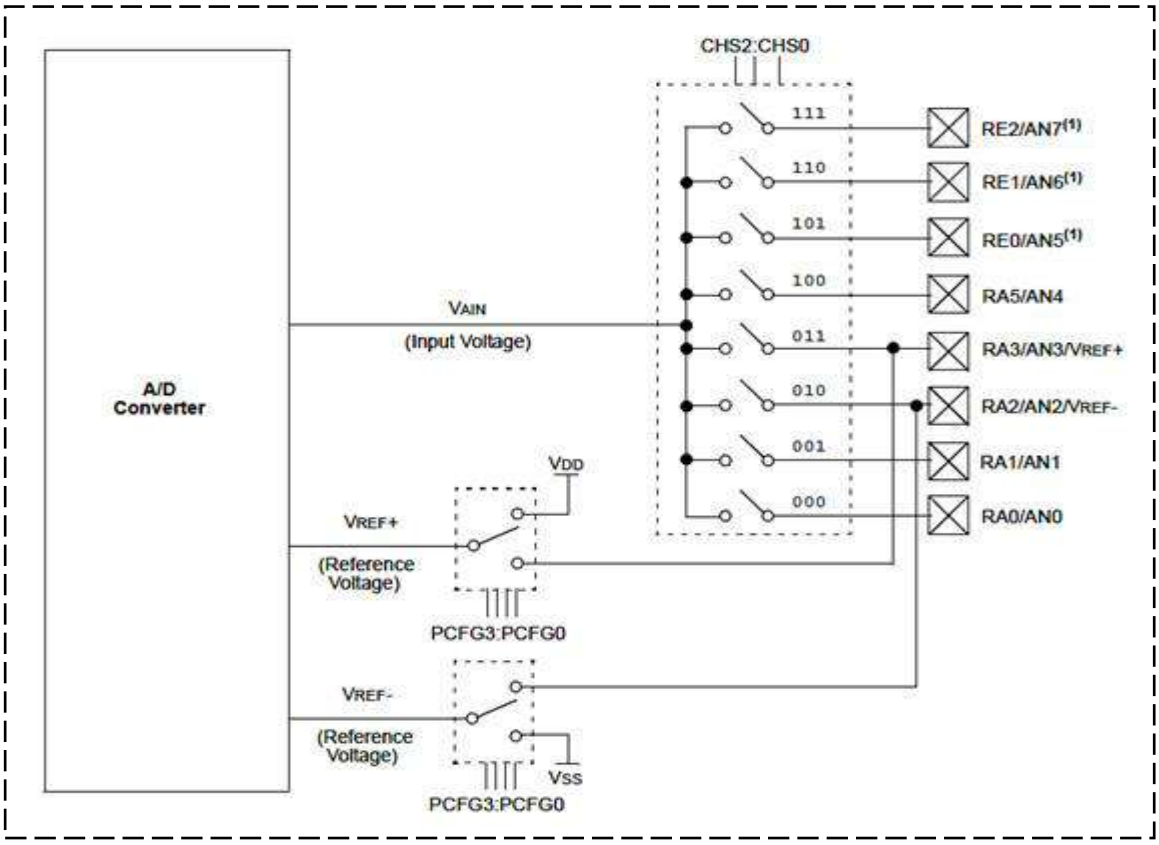


Figure I.23. *Schéma fonctionnel analogique numérique.* [1]

A. Les registres ADRESL et ADRESH

Il est important de savoir que le convertisseur donne un résultat sur 10 bits, et donc que ce résultat devra donc obligatoirement être sauvegardé dans 2 registres. Ces registres sont tout simplement les registres ADRESL et ADRESH. Comme 2 registres contiennent 16 bits, et que nous n'en utilisons que 10, Micro chip nous a laissé le choix sur la façon dont est sauvegardé le résultat, nous pouvons soit justifier le résultat à gauche, soit à droite.

La justification à droite complète la partie gauche du résultat par des « 0 ». Le résultat sera donc de la forme :

	ADRESH									ADRESL					
0	**0**	**0**	**0**	**0**	**0**	**B9**	**B8**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**

La justification à gauche procède bien évidemment de la méthode inverse :

	ADRESH								ADRESL						
B9	**B8**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	**0**	**0**	**0**	**0**	**0**	**0**

b. Le registre ADCON1

Ce registre permet de déterminer le rôle de chacune des pins AN0 à AN7. Il permet donc de choisir si un pin sera utilisé comme entrée analogique, comme entrée/sortie standard, ou comme tension de référence. Il permet également de décider de la justification du résultat.

Pour pouvoir utiliser un pin en mode analogique, il faudra que ce pin soit configuré également en entrée par TRISA. Le registre ADCON1 dispose, comme tout registre

Accessible de notre PIC, de 8 bits, dont seulement 5 sont utilisés :

B7	ADFM	A/D result format select
B6	INUTILISE	lu comme « 0 »
B5	INUTILISE	lu comme « 0 »
B4	INUTILISE	lu comme « 0 »
B3	PCFG3	Port Configuration control bit 3
B2	PCFG2	Port Configuration control bit 2
B1	PCFG1	Port Configuration control bit 1
B0	PCFG0	Port Configuration control bit 0

Tableau I.5. *Bits de registre ADCON1*

c. Le registre ADCON0

Ce registre est le dernier utilisé par le convertisseur analogique/numérique. Il contient les bits que nous allons manipuler lors de notre conversion. Sur les 8 bits de

Notre registre, 7 seront utilisés. [2]

B7	ADCS1	A/D conversion Clock Select bit 1
B6	ADCS0	A/D conversion Clock Select bit 0
B5	CHS2	Analog Channel Select bit2
B4	CHS1	Analog Channel Select bit1
B3	CHS0	Analog Channel Select bit0
B2	GO/DONE	A/D conversion status bit
B1	Inutilisé	lu comme « 0 »
B0	ADON	A/D ON bit

Tableau I.6. *Bits de registre ADCON0* [2]

I.6. LES AVANTAGES DES MICROCONTROLEURS

L'utilisation des microcontrôleurs pour les circuits programmables à plusieurs points forts et bien réels. Il suffit pour s'en persuader, d'examiner la spectaculaire évolution de l'offre des fabricants de circuits intégrés en ce domaine depuis quelques années.
Nous allons voir que nombre d'entre eux découle du simple sens.

- Tout d'abord, un microcontrôleur intégre dans un seul et même boîtier ce qui, avant nécessitait une dizaine d'éléments séparés. Il résulte donc une diminution évidente de l'encombrement en matériel et du circuit imprimé.
- Cette intégration a aussi comme conséquence immédiate de simplifier le tracé du circuit imprimé puisqu'il n'est plus nécessaire de véhiculer des bus d'adresses et de donnée d'un composant à un autre.
- L'augmentation de la fiabilité du système puisque, le nombre des composants diminuant, le nombre des connexions composants/supports ou composants/circuits imprimés diminue.
- Réalisation des applications non réalisables avec d'autres composants. [5]

I.7. CONCLUSION

Le présent chapitre constitue le cœur de notre projet. En effet, nous avons présenté L'architecture et les caractéristiques du microcontrôleur PIC 16f877a

En conclusion, nous pouvons dire que le microcontrôleur 16F877A peut bien jouer le rôle d'une unité de contrôle (commande) pour notre système, il contient tous les modules nécessaires pour la régulation. Pour fonctionner cette unité de contrôle il faut la programmer et l'adapter à un compilateur de programmation.

I.8 .machines à courant continu :

I.8 .1.introduction

L'évolution des technologies conduit à utiliser des machines nécessitant des vitesses de rotation précises et variables pour l'entraînement d'engins de manutention par exemple.

Une machine à courant continu est une machine électrique. Il s'agit d'un convertisseur électromécanique permettant la conversion bidirectionnelle d'énergie entre une installation électrique parcourue par un courant continu et un dispositif mécanique. Elle est aussi appelée dynamo.

En fonctionnement moteur, l'énergie électrique est transformée en énergie mécanique

En fonctionnement générateur, l'énergie mécanique est transformée en énergie électrique, la machine se comporte comme un frein.

I.8.2. ORGANISATION DE LA MACHINE

Dans l'organisation d'une machine à courant continu, on peut distinguer les principaux éléments suivants :

I.8.2.1 *Pôles inducteurs*

Les pôles inducteurs ont pour rôle de créer le flux inducteur dans la machine «entre la culasse et l'induit ».

Ce flux est généré soit par des enroulements, soit par des aimants.

Ce flux est canalisé dans la machine par des matériaux ferromagnétiques «parties hachurées ».

Ce flux étant constant dans la partie portant les pôles inducteurs et dans les pôles inducteurs eux-mêmes.

I.8.2.2 *Stator ou inducteur*

Le stator partie fixe, constitué par un aimant ou un électroaimant appelé également inducteur qui crée un champ magnétique dirigé vers l'axe du rotor. Il est formé d'aimant figure (1.1-a) pour les petits moteurs et des bobines figure (1.1-b) parcourue par un courant continu. Les aimants ou bobines permettent la création d'un flux magnétique.



Figure I.24. ***a*** *: aiment,* ***b*** *: bobines* **[6]**

I.8.2.3 *Induit « rotor »*

Le rotor, partie mobile, appelé également induit, constitué par un cylindre d'acier doux à la périphérie du quel sont disposés des conducteurs reliés aux lames du collecteur sur lesquelles frottent deux charbons, ou balais, qui assurent la liaison avec les bornes du moteur.

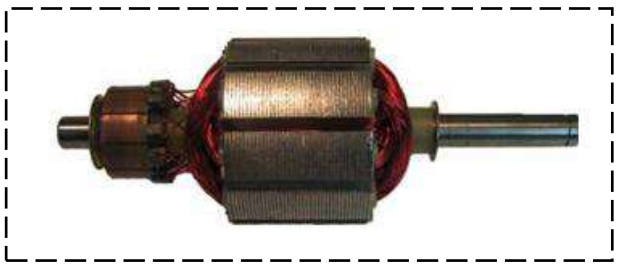


Figure I.25. *Induit* **[6]**

I.8.2.4 *Collecteur*

Il est formé d'un ensemble de lames de cuivre isolées latéralement les unes des autres et disposées suivant un cylindre en bout de rotor. Deux balais portés par le stator frottent sur les lames du collecteur**. [6]**

Prenons un cas particulier d'une machine à courant continu à vitesse variable dont la réaction du courant d'induit et du champ inducteur crée un couple électromagnétique **C** donné par la formule :

$$C = K \varphi i \qquad \text{(I.1)}$$

Avec $k = \left(\frac{2p_1}{2a}\right)\frac{n}{2\pi}$ coefficient de couplage électromécanique,

$2p_1$: nombre de pôles,

2a : nombres de voie d'enroulement,

n : nombre de conducteurs de l'induit.

La conservation de l'énergie appliquée à l'induit

$$C\Omega = e\ i \qquad \textbf{(I.2)}$$

Avec e force contre-électromotrice (f.c.é.m.) due à la vitesse angulaire de rotation Ω du rotor.

Cette relation n'a de sens que dans un système normalisé (Ω en rad/s).

Par suite, il vient :

$$e = k\ \varphi\ \Omega \qquad \textbf{(I.3)}$$

L'équation électrique relative à l'induit est : relative à l'induit est

$$v = Ri + L\frac{di}{dt} + e = Ri + L\frac{di}{dt} + k\ \varphi\ \Omega \qquad \textbf{(I.4)}$$

Avec L et R inductance propre et résistance de l'induit.

Dans le cas des machines excitées par un courant i_f, il est nécessaire de connaitre la caractéristique magnétique **(Fig.I.26)**

$$\varphi = F\ (i_f) \qquad \textbf{(I.5)}$$

L'équation relative à l'inducteur, dont les n_f spires ont une résistance R_f, est :

$$V_f = R_f\ i_f + n_f\ \frac{d\varphi}{dt} \qquad \textbf{(I.6)}$$

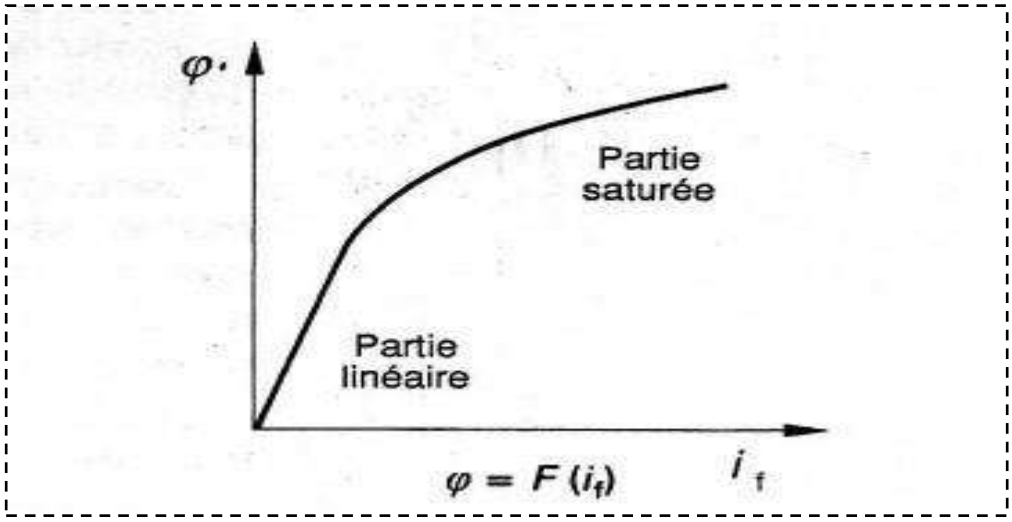


Figure I.26. *Caractéristique magnétique d'une machine à courant continu* **[1]**

Quand on réalise cette équation, on définit l'inductance propre $\mathbf{L_f}$ de l'inducteur **:**

$$\mathbf{L_f = n_f \frac{d\varphi}{dt}} \qquad \textbf{(I.7)}$$

La figure I.4 représente un schéma équivalent de la machine, dont le paragraphe **1.7** donnera le modèle dynamique complet**. [1]**

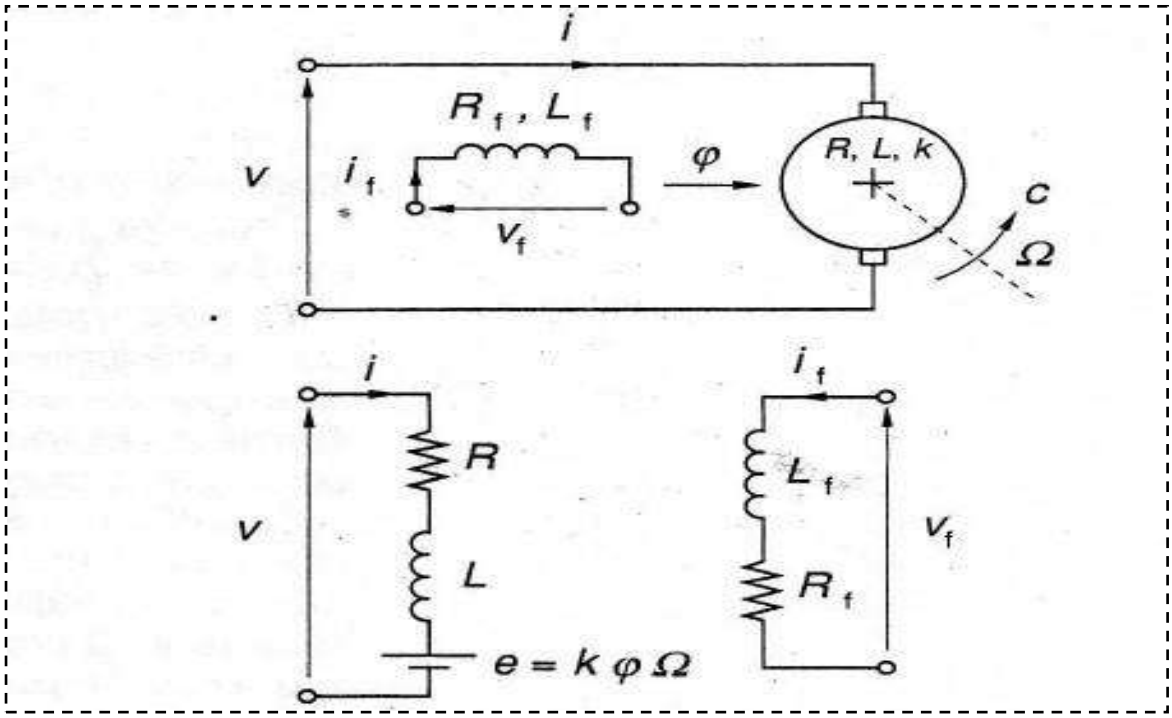


Figure I.27. *Schéma équivalent du moteur à CC.* **[7]**

I.8.3 .CARACTERISTIQUES MECANIQUES ET ELECTRIQUES

On constate, voir figure (1.5), que la fréquence de rotation n diminue lorsque le couple C augmente « à tension U constante ».

Le graphe de droite montre que la fréquence de rotation est proportionnelle à la tension, à couple constant.

Remarque : Pour inverser le sens de rotation, on inversera le courant dans l'induit « on pourrait aussi inverser les courant inducteur, mais cela n'est pas possible si le flux inducteur est créé par des aimants permanents ». **[6]**

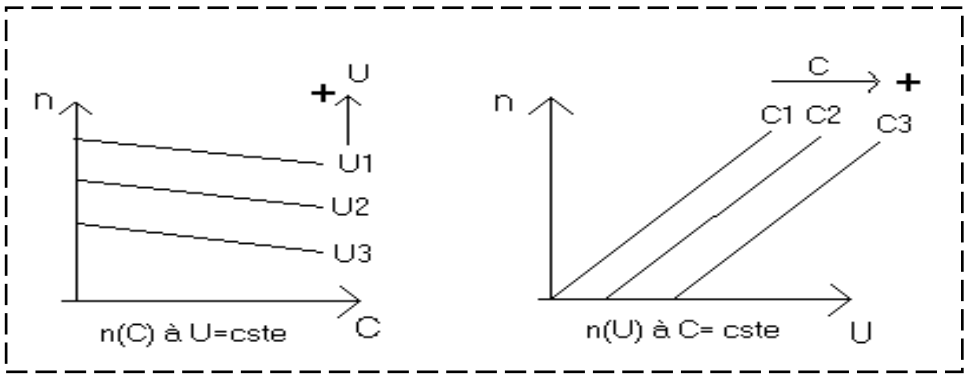


Figure I.28. *Caractéristiques mécaniques et électriques* **[7]**

I.8.4. LES DIFFERENTS TYPES DE MOTEURS

On peut réaliser l'excitation des moteurs de plusieurs façons :

a. En dérivation : C'est une excitation séparée, l'inducteur comporte beaucoup de spires de faibles sections.

b. En série : L'inducteur est traversé par le courant d'induit, il comporte peut de spires de grosses sections.

c. Composé : On trouve une partie des enroulements inducteurs en série et une partie en parallèle. On distingue deux types de branchements :

c.1 Courte dérivation : L'enroulement parallèle est aux bornes du moteur.

c.2 Longue dérivation : L'enroulement parallèle est aux bornes de la source d'alimentation.

On peut réaliser pour chaque type de branchement deux types d'excitations :

Une excitation à flux additif, le flux de l'enroulement série s'ajoute avec celui de l'enroulement parallèle.

Une excitation à flux soustractif, le flux série se soustrait de celui de l'enroulement parallèle.

Le type d'excitation a une influence sur le comportement du moteur. **[6]**

I.8.5. ORGANES MECANIQUES

Le stator autoporteur reçoit de chaque côté un flasque sur lequel le rotor sera positionné grâce à des roulements à billes ou à rouleaux suivant le type de charge « axiale ou radiale ». Les pôles inducteurs sont en général vissés peuvent aussi en faire partie intégrante ».

Un ventilateur est placé en bout d'arbre, sur le rotor, pour le refroidissement de la machine, il peut être complété par une ventilation forcée motorisée pour le refroidissement aux vitesses lentes. **[6]**

Le collecteur, l'ensemble porte-balais et les balais se situent de l'autre côté de la machine.

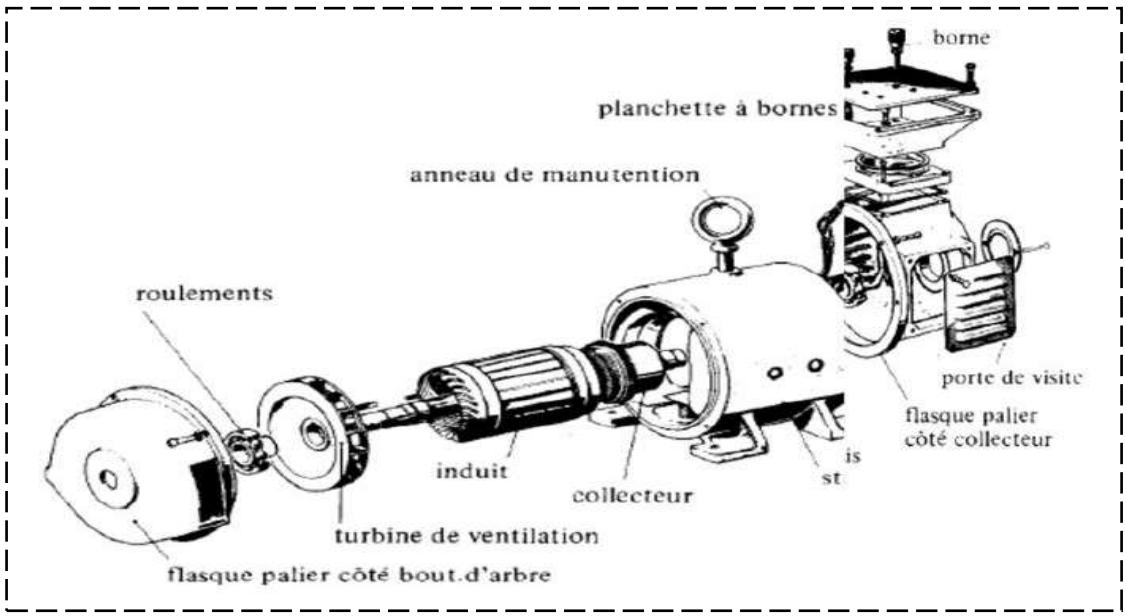


Figure I.28. *Constitution de la MCC* **[6]**

I.8.6. PRINCIPE DE FONCTIONNEMENT

Lorsque les conducteurs sont parcourus par un courant, ils sont soumis à des forces F1 et F2 qui tendent à faire tourner le rotor. Le collecteur permet d'inverser le sens du courant dans les conducteurs lorsque ceux-ci passent le plan vertical. Ainsi le sens du couple des forces F1 et F2 et donc le sens de rotation du moteur est conservé. **[2]**

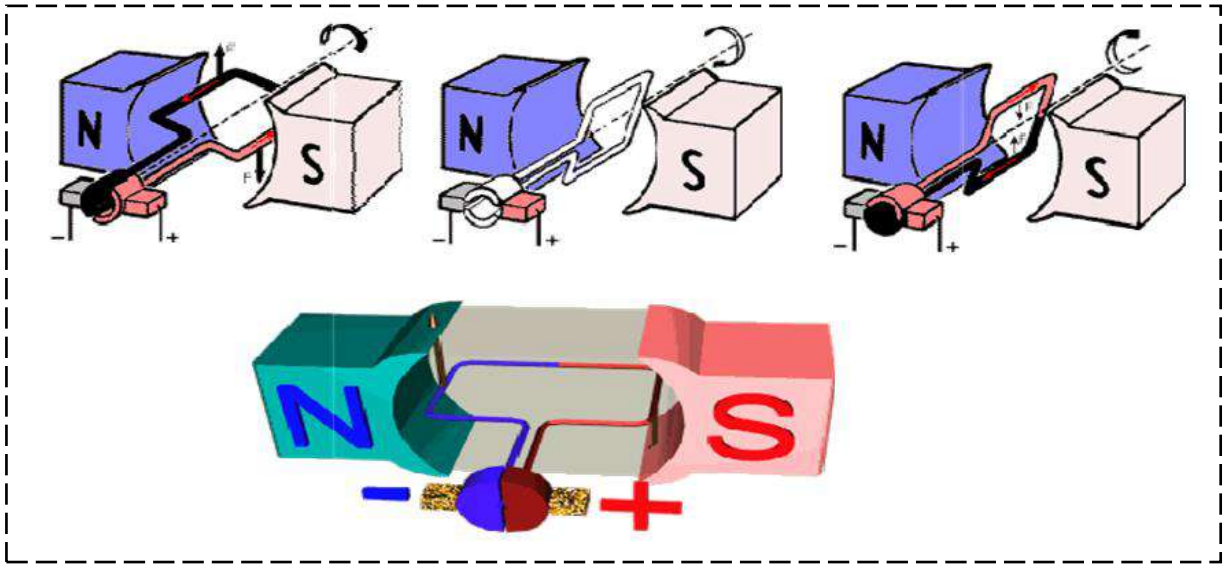


Figure I.29. *Principe de fonctionnement* **[6]**

I.8.7. FORCE ELECTROMOTRICE

Nous savons qu'une bobine en mouvement dans un champ magnétique voit apparaître à ses bornes une force électromotrice « **f.é.m**. » donnée par la loi de Faraday, sur ce principe, la machine à courant continu est le siège d'une f.é.m. E :

$$E=\frac{p}{2\pi a} N \phi \Omega \qquad \textbf{(I.8)}$$

I.8.8 .FONCTIONNEMENTS

Imaginons une machine électrique alimentée par une source de tension U constante. Lorsque le moteur tourne à vide il n'a pas besoin de fournir de couple, $I_{\underline{a}}$ est très faible et $U \approx E$. La vitesse de rotation est proportionnelle à U.

I.8.8.1 fonctionnement en moteur

Lorsque l'on veut la faire travailler, en appliquant un couple résistant sur son axe, cela la freine donc E diminue. Comme U reste constante, le produit $R_i.I_i$ augmente donc I_i augmente, donc le couple T augmente lui aussi et lutte contre la diminution de vitesse : c'est un ***couple moteur***.

Plus on le freine, plus le courant augmente pour lutter contre la diminution de vitesse. C'est pourquoi les moteurs à courant continu peuvent « griller » lorsque le rotor est bloqué, si le courant de la source n'est pas limité à une valeur correcte.

I.8.8.2 fonctionnement en génératrice

Si une source d'énergie mécanique essaie d'augmenter la vitesse de machine, « la charge est entraînante : ascenseur par exemple », Ω augmente donc E augmente. Comme U reste constante, le produit $R_i.I_i$ devient négatif et augmente en valeur absolue, donc I_i augmente, donc le couple T augmente lui aussi et lutte contre l'augmentation de vitesse : c'est un ***couple frein***.

Le signe du courant ayant changé, le signe de la puissance consommée change lui aussi. La machine consomme une *puissance négative*, donc elle fournit de la puissance au circuit. Elle est devenue génératrice. **[6]**

Ces deux modes de fonctionnement existent pour les deux sens de rotation de la machine. Celle-ci pouvant passer sans discontinuité d'un sens de rotation ou de couple à l'autre. On dit alors qu'elle fonctionne dans les quatre quadrants du plan couple vitesse.

I .8.8.3 Bilan des puissances

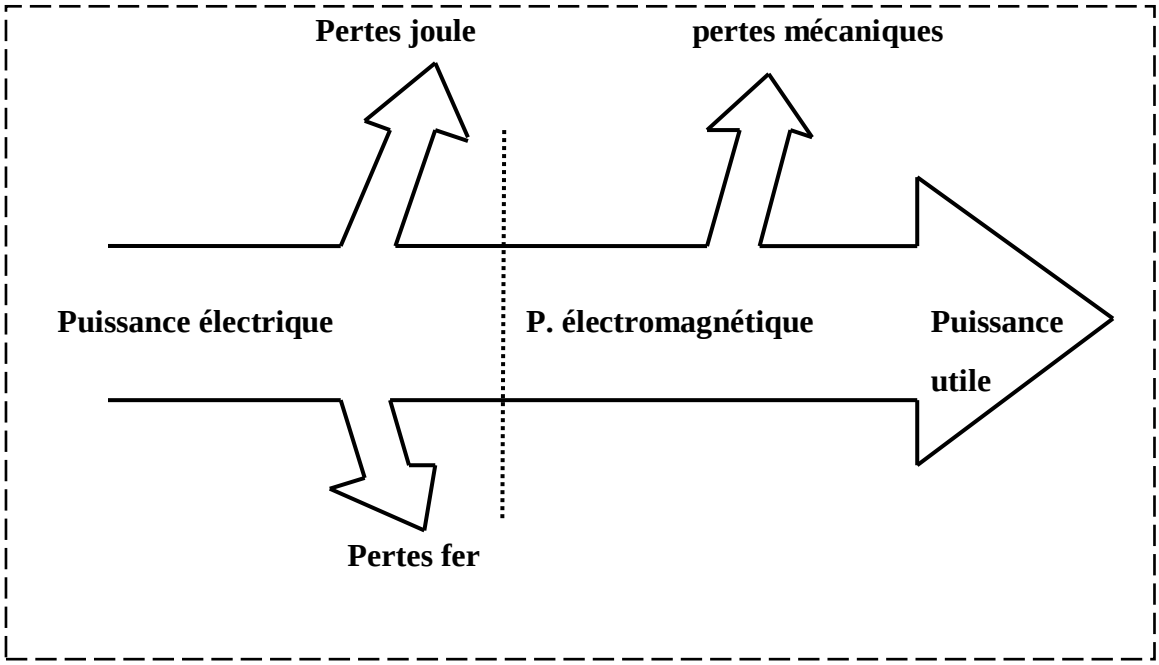


Figure I.30. *Bilan des puissances d'un moteur à courant continu* **[6]**

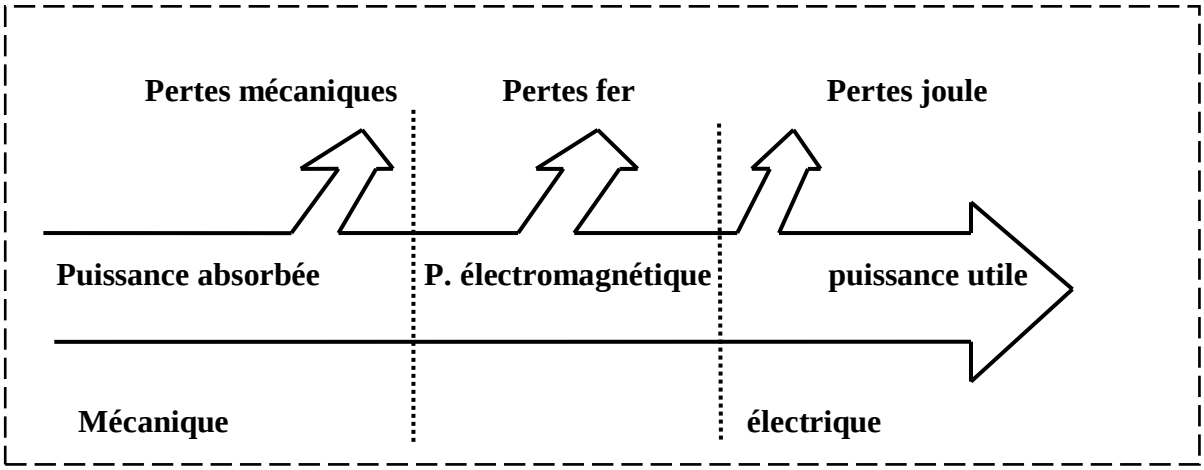


Figure I.31. *Bilan des puissances d'un générateur à courant continu* **[6]**

a. Puissance utile « sur l'arbre en sortie moteur » :

P_u = P_{em} – « pertes mécaniques + pertes magnétiques », en Watts

b. Puissance absorbée « en entrée » :

P_{abs} = U.I « U : tension d'alimentation en Volt, I : courant absorbé en ligne, P_{abs} en Watts»

c. Pertes :

Pertes par effet joules « $R.I^2$ », pertes magnétiques, et pertes mécaniques

d. Rendement :

Le rendement = Puissance fournie « utile » / Puissance totale absorbée. = P_u / P_{abs}

P_{abs} = U.I « + puissance absorbée par l'inducteur » P_u = C_u. Ω**,** d'où rendement = (C_u* Ω)/UI

I.8.9. VARIATION DE LA VITESSE DE LA MACHINE A COURANT CONTINU

$$\Omega = u\ Ra\ Ia/k \qquad \textbf{(I.9)}$$

La figure suivante représente : **a :** action sur la tension à flux constant- la figure représente la courbe de variation de la vitesse en fonction de la tension pour une charge constante.

B : action sur le flux à tension constante, la figure représente la courbe de variation de la vitesse en fonction du flux d'excitation pour une charge constante. **[7]**

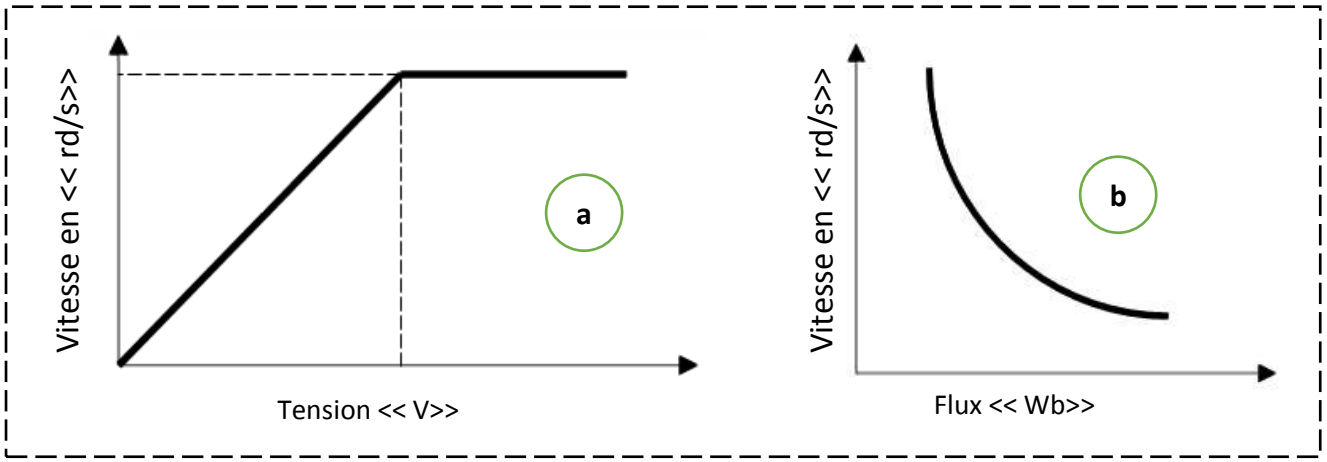


Figure.32. ***a****. Variation de la vitesse en fonction de la tension d'induit* ***b****. Variation de la vitesse en fonction du flux d'inducteur* **[7]**

I.8.10. PROCEDE DE COMMANDE D'UN MOTEUR A COURANT CONTINUE

L'expression générale de la vitesse montre les trois paramètres R_a, R_{reg} et U que l'on peut régler ce qui donne les trois procédés pour la variation s de vitesse :

I.8.10.1 Réglage rhéostatique

Il est possible de réduire la vitesse en augmentant la résistance de l'induit avec un rhéostat branchée en série avec l'induit tout en fixant la tension et le flux à leurs valeurs nominales, l'expression de la vitesse est :

$$\Omega = \frac{U\ Ra\ Rreg\ Ia}{K} \qquad \textbf{(I.10)}$$

Ce type de réglage est mauvais du point de vue rendement. Le rhéostat insère a une consommation d'énergie d'autant plus importants que la chute de vitesse est élevée, c'est ainsi que ce procède de réglage n'est utilisé que pour le démarrage et le freinage.

I.8.10.2 Réglage par le flux

Au démarrage, un moteur doit absorber un courant important pour pouvoir porter à la vitesse nominale toutes les masses et charges qui sont initialement au repos, d'où la nécessite d'appliquer un flux maximum pour permettre au moteur de développer une force électromotrice d'une valeur suffisante et d'une manière rapide.

Ce type de réglage est bon du point de vue technique, et également du point de vue économique car la puissance dissipée dans l'inducteur étant très faible par rapport à la puissance absorbée, le rendement du moteur ne sera pas modifier.

I.8.10.3 Réglage par tension

On applique une tension de valeur moyenne variable entre zéro et une valeur maximale tout en réglant le flux à sa valeur maximale. Ce procédé de réglage de vitesse est excellent du point de vue économique car aucune énergie n'est dissipe dans les rhéostats. Le réglage par la tension permet seulement de diminuer la vitesse.

I.8.11. MODELISATION EN REGIME DYNAMIQUE

I.8.11.1 Fonction de transfert du moteur à courant continu à excitation séparée

En vitesse variable, la machine est associée à des convertisseurs et à des régulateurs dont il faut déterminer les paramètres : c'est la synthèse du système, qui ne peut se faire raisonnablement que dans le cadre du système linéaire continu, à coefficients constants. Or, au paragraphe 1,61, nous avons vu qu'une analyse physique précise conduit à des modèles non linéaires. Pour simplifier les équations, on ne représente pas les effets de la saturation ou de la réaction d'induit. On considère que l'inductance de l'induit **L** est constante et onregroupe, en une expression du premier degré ($f\,\Omega+C_0$), l'ensemble des couples de frottement et du couple de charge. On se ramène à des équations différentielles linéaires à coefficients constants. Le modèle dynamique complet est défini par :

$$\mathbf{V = R\,i + L\,\frac{di}{dt} + K\,\Omega} \qquad \textbf{(I.11)}$$

$$\mathbf{C = K\,i = J\,\frac{d\Omega}{dt} + f\,\Omega + C_0} \qquad \textbf{(I.12)}$$

Avec $\mathbf{C_0}$ **:** couple de charge,

f : coefficient de frottement visqueux,

j : inertie des parties tournantes.

Il est fréquent (mais il faut le vérifier dans chaque cas) que les paramètres **L** et **f** soient négligeables. Par ailleurs, on ne peut pas s'intéresser à l'effet de $\mathbf{C_0}$ (par exemple quand le moteur est à vide) ; aussi, selon que l'on néglige **L**, **f** ou $\mathbf{C_0}$, on obtient huit fonctions de transfert possibles, correspondant à des niveaux de simplification différents

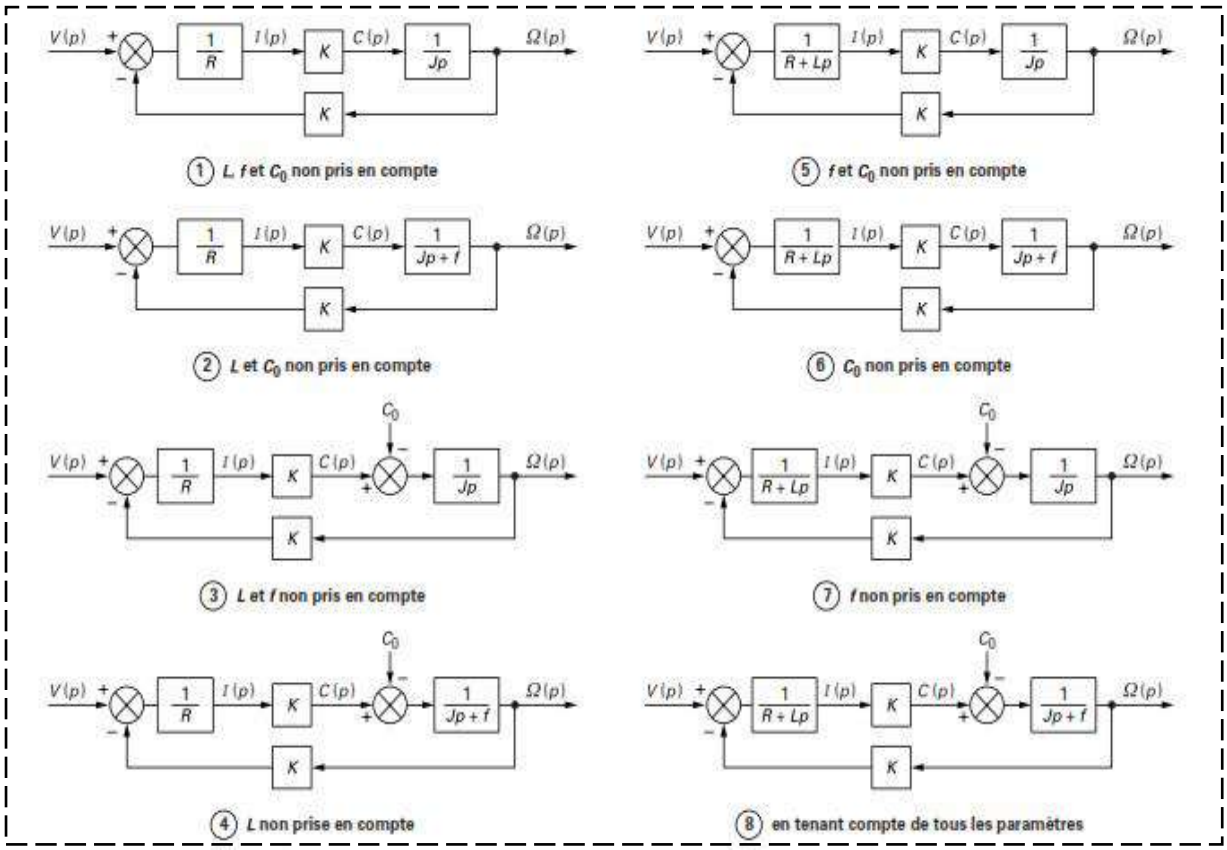


Figure I.33. *Classification des huit fonctions de transfert du moteur à courant continu (obtenus en négligeant L, f ou C0).* **[7]**

On reconnait, dans le modèle -1-, le modèle le plus simple qui nous a servi à définir les caractéristiques dynamiques des servomoteurs. A l'opposé, le modèle le plus complet est le modèle -8- déduit sans simplification des relations I.12 et I.13 .La fonction de transfert a pou expression complète :

$$\frac{\Omega(p)}{V(p)} = \frac{Km}{1+(\tau_{em}+\mu\tau_e)p+\tau_e\tau_{em}p^2} \qquad \textbf{(I.134)}$$

Avec $K_m = K/ (K^2+R\ f)$ gain statique,

$\mu = R\ f/(K^2+R\ f)$ coefficient habituellement petit,

$\boldsymbol{\tau_e = L/R}$ Constante de temps électrique ;

$\boldsymbol{\tau_{em} = \frac{RJ}{K^2+Rf}}$ Constante de temps électromécanique, habituellement (mais improprement) appelée constante de temps mécanique dans la littérature technique, ou on la note$\boldsymbol{\tau_m}$.

Le plus souvent, mais pas toujours, les pôles de la fonction de transfert sont réels. On peut alors définir deux constantes de temps$\boldsymbol{\tau'}$ **et** $\boldsymbol{\tau''}$**.** On obtient alors :

$\mu=0$

$K_m=1/K$

$\boldsymbol{\tau' \approx \tau_e}$

$\boldsymbol{\tau'' \approx \tau_{em} = R\ J/K^2}$

Dans ces conditions, la fonction de transfert est habituellement écrite :

$$\frac{\Omega(p)}{V(p)} = \frac{1}{K}\ \frac{1}{(1+\tau_e p)(1+\tau_{em} p)} \qquad \textbf{(I.14)}$$

I.8.11.2 *Fonction de transfert du moteur en régime sous-excité*

Pour l'étude des régulations, il est naturel de linéariser ce modèle autour d'un point de fonctionnement défini par$(\boldsymbol{\Omega_0, i_0, \Phi_0})$**,** l'inductance propre de l'inducteur étant définie par la relation (6) :

$$L_f = n_f \frac{d_\varphi}{di_f} \quad \textbf{(I.15)}$$

I.8.12. AVANTAGES ET INCONVENIENTS

L'avantage principal des machines à courant continu réside dans leur adaptation simple aux moyens permettant de régler ou de faire varier leur vitesse, leur couple et leur sens de rotation : les ***variateurs de vitesse*.** Voire leur raccordement direct à la source d'énergie : batteries d'accumulateur, piles, etc.

Le principal problème de ces machines vient de la liaison entre les balais, ou « Charbons » et le collecteur rotatif. Ainsi que le collecteur lui-même comme indiqué plus haut et la complexité de sa réalisation. De plus il faut signaler que :

Plus la vitesse de rotation est élevée, plus la pression des balais doit augmenter pour rester en contact avec le collecteur donc plus le frottement est important.

Aux vitesses élevées les balais doivent donc être remplacés très régulièrement.

Le collecteur imposant des ruptures de contact provoque des arcs, qui usent rapidement le commutateur et génèrent des parasites dans le circuit d'alimentation, ainsi que par rayonnement électromagnétique.

Un autre problème limite les vitesses d'utilisation élevées de ces moteurs lorsque le rotor est bobiné, c'est le phénomène de « dé frettage », la force centrifuge finissant par casser les liens assurant la tenue des ensembles de spires « le frettage ».

Un certain nombre de ces inconvénients ont partiellement été résolus par des réalisations de moteurs sans fer au rotor, comme les moteurs « disques » ou les moteurs « Cloches », qui néanmoins possèdent toujours des balais.

Les inconvénients ci-dessus ont été radicalement éliminés grâce à la technologie du moteur *brushless*, aussi dénommé « moteur à courant continu sans balais », ou moteur sans balais. **[6]**

I.8.13. *Encodeur incrémental*

I.8.15. 1. Principe:

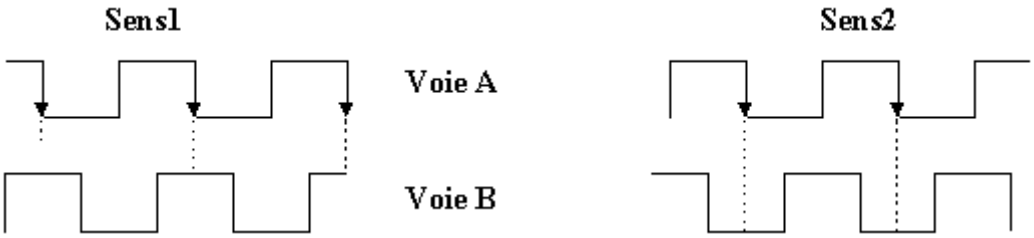
Une lumière émise par une diode électroluminescente est réfléchie par les graduations d'un disque vers un phototransistor qui se sature et se bloque à la cadence du défilement des graduations.



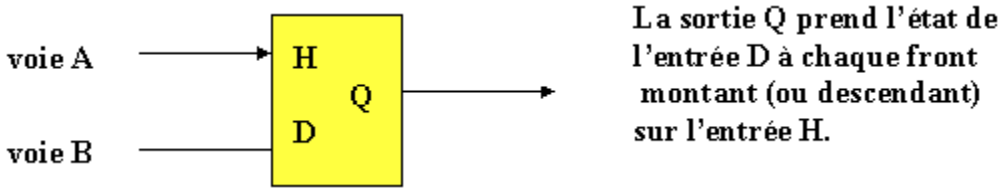
Figure I.34. Encodeur incrémental

Le codeur incrémental est surtout utilisé dans les systèmes dont le traitement de l'information est entièrement numérique. Ses impulsions sont comptabilisées de façon à donner une information concernant la position (nombre d'impulsions délivrées depuis une position d'origine) ou /et une information concernant la vitesse (nombre d'impulsions par unité de temps). Dans ce dernier cas, il évite l'emploi d'une génératrice tachymétrique (il est cependant peu précis aux très basses fréquences de rotation).

Un codeur incrémental possède généralement plusieurs voies:
- voie Z donnant une impulsion par tour,
- voie A donnant n impulsions par tour,
- voie B identique à voie A, mais dont les signaux sont déphasés de + ou - 90°, suivant le sens de rotation.



La simple utilisation d'une bascule D permet d'obtenir une information logique concernant le sens de rotation:



Caractéristiques principales d'un encodeur incrémental :

- nombre de points par tour ou nombre d'impulsions par tour (exemple: 500 points/tr),
- nombre de voies ou nombre de pistes (exemple: 6 pistes A, B, Z, A*, B*, Z*),
- tension d'alimentation,
- vitesse maximale de rotation.

I.8.15. 2.Résume:

Certains signaux peuvent être convertis du domaine continu discret sans utiliser de convertisseur A/N. la position du rotor d'un moteur en est un exemple. En effet il existe plusieurs capteurs dédiés à mesurer la position du rotor d'un moteur directement dans le domaine numérique. Le moins dispendieux de ces capteurs est l'encodeur incrémental.

L'encodeur incrémental peut mesurer le déplacement angulaire du rotor d'un moteur électrique. Le principe de fonctionnement de ce capteur est illustré par la figure I.33 le disque en rotation est branché mécaniquement avec le rotor du moteur. Ainsi, parce que les disques renferment des régions opaques et des régions transparentes, selon la position du rotor, les faisceaux lumineux peuvent atteindre les senseurs photoélectriques. Les signaux fournis par ces senseurs sont par conséquents parfois à une valeur maximum lorsque la lumière passe et parfois à une valeur minimum lorsque la lumière est bloquée. Connaissant le nombre de régions transparentes sur les disques, le déplacement du rotor peut être obtenu en comptent le nombre de valeurs maximales du signal A ou du signal B. [8]

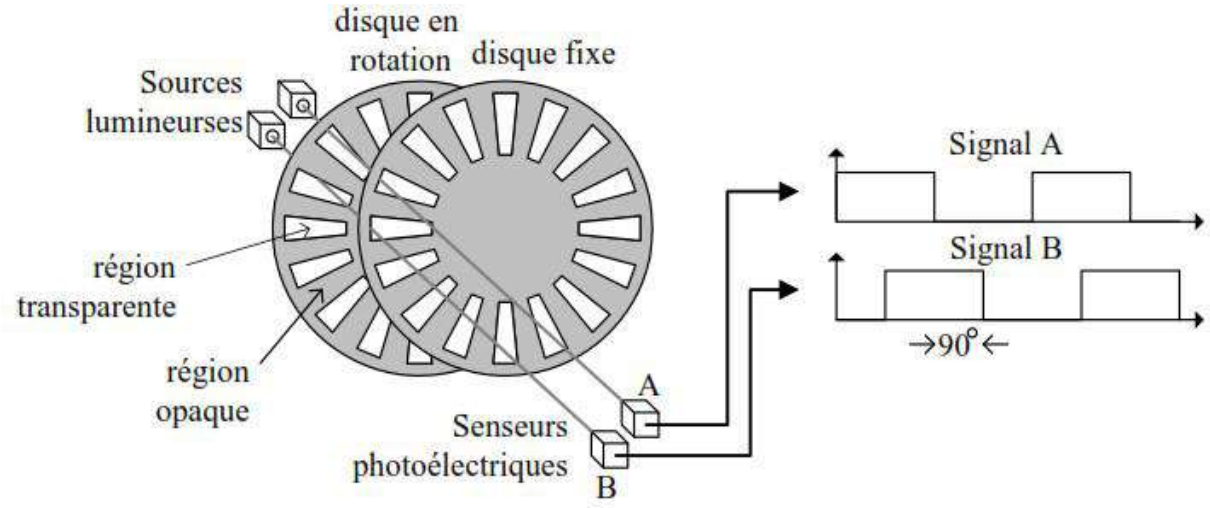


Figure I.35. *Encodeur incrémental.* [8]

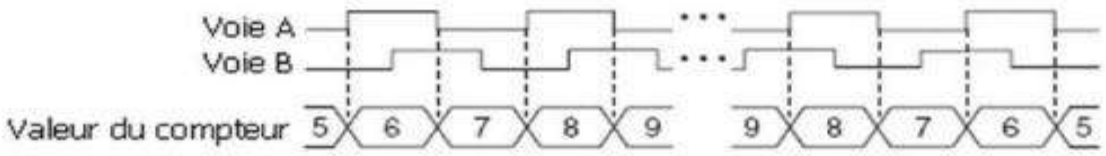


Figure I.36. *Encodage X2.* [9]

La figure I.34 représente un cycle de quadrature et les incréments et décréments résultants pour l'encodage X2, [9]

La figure I.30 représente les sources lumineuses et senseurs A et B sont placés de façon à ce que les signaux provenant de ces capteurs soient en quadrature ; c'est-à-dire, déphasés de 90 degrés. Cette particularité nous permet de déterminer le sens, le signal B est en avance sur le signal A tandis que lorsqu'il tourne dans l'autre sens, B est en retard sur A. le décodage des signaux A et B peut donc se faire en utilisant un compteur branché sur le signal A ou B qui s'incrémente si B est en avance sur A et qui se décrémente si B est en retard sur A. pour augmenter la précision de la mesure par un facteur de quatre, il est également possible de compter tous les transitions des signaux A et B. [8]

Le désavantage de l'encodeur incrémentale est que la position mesurée est toujours relative à la position du rotor du moteur au moment où le système a subi une remise à zéro. Pour contourner ce problème, il est fréquent que l'encodeur incrémentale possède un troisième signal produisant une impulsion une seule fois par tour. Ce signal peut alors nous permettre de

prévoir un mécanisme d'initialisation permettant de placer le rotor à une position connue lors de la remise à zéro. Il existe également d'autre type d'encodeur permettant de fournir directement la position du rotor du moteur et non son déplacement. Ces encodeurs beaucoup plus dispendieux sont appelés encodeurs absolus. [8]

Le gain de capteur est alors donné par :

$$K_E = 4\, N_t \frac{1\ tour}{2\ \pi\ rad} \qquad \textbf{(I.16)}$$

Où :

N_t : est le nombre total de régions transparentes sur les disques de l'encodeur incrémental.

Pour les positions rotatives :

$$\theta = \frac{nombre\ de\ fronts}{x\ N} * 360 \qquad \textbf{(I.17)}$$

Où : N = nombre d'impulsions générées par l'encodeur à chaque révolution de l'axe

x = type d'encodage

I.9. CONCLUSION

La machine à courant continu a été durant de nombreuses années l'actionneur principalement utilisé dans les applications à vitesse variable. En effet, comme cela a été mis en évidence, le contrôle de la vitesse de rotation peut être aisément réalisé par action sur la f.é.m. d'induit de la machine « en grandeur permanente, donc moyenne ». Sur un autre plan, on peut agir sur le flux inducteur qui est réglé par le courant dans l'inducteur « appelé aussi courant d'excitation ».

Chapitre II :

La régulation analogique

II. INTRODUCTION

Malgré tous les développements théorique de la commande depuis les 50 dernières années, les compensateurs proportionnel intégrale (PI), proportionnel dérivée (PD), et proportionnel intégral dérivée (PID) demeurent aujourd'hui les plus utiliser en milieu industriel.

Dans cette section, nous allons d'abord développer le modèle discret des compensateurs PI. Nous allons ensuite discuter à l'encodeur incrémental et finalement on voir la régulation analogique de la MCC à travers le compensateur PI.

II.1.Principe général de la régulation

Dans la plupart des appareils et installations industrielles, tertiaires et mêmes domestiques, il est nécessaire de maintenir des grandeurs physiques à des valeurs déterminées, en dépit des variations externes ou internes influant sur ces grandeurs,

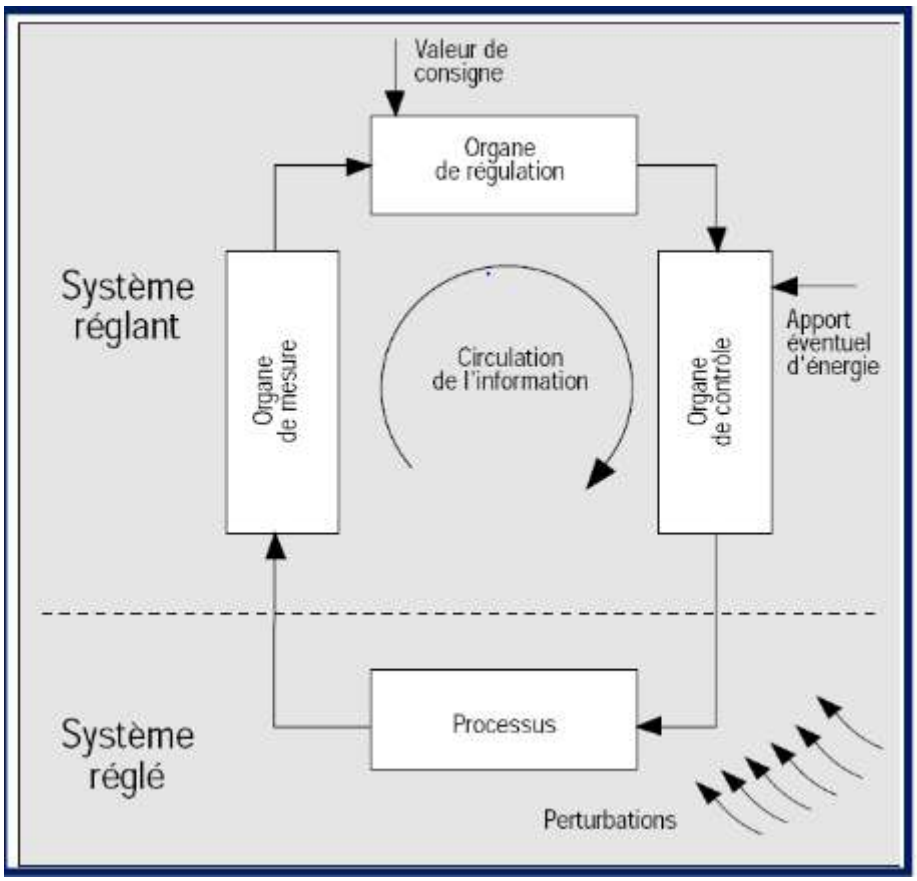


Figure. II.1. *Schéma de principe d'une chaine de régulation. [10]*

II.2 .Les actions PID :

En pratique, à une catégorie donnée de systèmes à asservir correspond un type de correcteur adopte. Pour effectuer un choix judicieux, il faut connaitre les effets des différentes actions : proportionnelle, intégrale et dérivée.

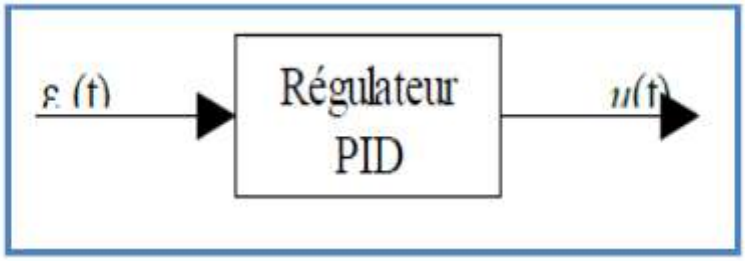


Figure. II.2. *Schéma synoptique d'un régulateur PID. [10]*

Un régulateur PID est obtenu par l'association de ces trois actions et il remplit essentiellement les trois fonctions suivantes :

1. Il fournit un signal de commande en tenant compte de l'évolution du signal de sortie par rapport à la consigne.
2. Il élimine l'erreur statique grâce au terme intégrateur
3. Il anticipe les variations de la sortie grâce au terme dérivateur.

La commande U(t) donnée par le régulateur PID, dans sa forme classique est décrite par :

$$\cup(t) = kp[\varepsilon(t) + \frac{1}{Ti}\int_0^t \mathcal{E}(\tau)d(\tau)\,d\tau + Td\ \frac{d\varepsilon(t)}{dt} \qquad \textbf{(II.1)}$$

Elle est la somme détroits termes :

- Le terme proportionnel :

$$P = Kp\ \varepsilon(t) \qquad \textbf{(II.2)}$$

- Le terme intégral :

$$I = Kp\ \frac{1}{Ti}\int_0^t E(\tau)d\tau \qquad \textbf{(II.3)}$$

- Le terme dérivatif :

$$D = Kp\ Td\ \frac{d\varepsilon(t)}{dt} \qquad \textbf{(II.4)}$$

Les paramètres du régulateur PID sont le gain proportionnel Kp, le temps intégral Ti et le temps dérivatif Td, les temps étant exprimés en secondes.

II.2 .1.L'action proportionnelle :

- La sortie U(t) du régulateur proportionnel est donnée en fonction de son entrée Ɛ(t) qui représente l'écart entre la consigne et la mesure par la relation :

$$\boldsymbol{U(t) = k_{p.}\mathcal{E}(t)} \qquad \textbf{(II.5)}$$

- Pour le cas discret, cette relation reste la même telle que :

$$\boldsymbol{U(t) = K_{p}.\mathcal{E}(K)} \qquad \textbf{(II.6)}$$

Le rôle de l'action proportionnelle est de minimiser l'écart Ɛ entre la consigne et la mesure et elle réduit le temps de montée et le temps de réponse. On constate qu'une augmentant du gain Kp du régulateur entraine une diminution de l'erreur statique et permet d'accélérer le comportement global de la boucle fermée. Oserait tenter de prendre des Valeurs de gain élevées pour accélérer la réponse du procédé Mais on est li mité par la stabilité de la boucle fermée. En effet, une valeur Trop élevée du gain augmente l'instabilité du système et donne lieu à des oscillations. [10]

II.2 .2.L'action dérivée :

Elle est une action qui tient compte de la vitesse de variation de l'écart entre la consigne et la mesure, elle joue aussi un rôle Stabilisateur, contrairement à l'action intégrale.
En effet, elle délivre une sortie variant proportionnellement à la vitesse de variation de l'écart.

$$\boldsymbol{U(t) = T_{d}\frac{d_{\mathcal{E}}(t)}{dt}} \qquad \textbf{(II.7)}$$

Avec Td le dosage de l'action dérivée, exprimée minutes ou en secondes. L'action dérivée va ainsi intervenir uniquement sur la variation de l'erreur ce qu'augmente la rapidité du système (diminution des temps de réponses). L'action dérivée permet aussi d'augmenter la Stabilité du système par apport de phase (+π/2ce qui augmente la marge de phase).
L'annulation de cette action en régime statique impose donc de ne jà Mais l'utiliser seule : L'action dérivée n'exerce qu'un complément à l'action proportionnelle.
En pratique, il est souhaitable de limiter l'action dérivée afin de ne pas amplifier les bruits haute fréquence et de limiter l'amplitude des impulsions dues aux discontinuités de l'écart.
Lorsque la période d'échantillonnage Te est petite, la différence vers l'arrière (Approximation d'Euler rétrograde) nous permet d'approcher la dérivée d'un signal à temps continu par :

$$\boldsymbol{\frac{df(t)}{dt} = \frac{f(t)-f(t-T_{e})}{T_{e}}} \qquad \textbf{(II.8)}$$

Dans le domaine de *Laplace*, l'opération de dérivation se traduisait par une multiplication par La variable de Laplace $\boldsymbol{p}$ en continu. Dans le cas discret, en appliquant la transformée en $\boldsymbol{z}$ à l'équation (I. 12) on obtient :

$$\boldsymbol{Z}\left\{\frac{f(t)-f(t-T_e)}{T_e}\right\} = \frac{1-z^{-1}}{T_e}\boldsymbol{F(z)} \qquad \textbf{(II.9)}$$

Ceci conduit à établir l'équivalence linéaire entre la variable de *Laplace* $\boldsymbol{p}$ et la variable

$$\boldsymbol{P \rightarrow \frac{1}{T_e}.(1-z^{-1})} \qquad \textbf{(II.10)}$$

Donc en discret, le terme dérive peut être remplace par :

$$\boldsymbol{\mathcal{U}(k) = \frac{T_d}{T_e}.(\mathcal{E}(k) - \mathcal{E}(k-1))} \qquad \textbf{(II.11)}$$

II.2 .3.L'action intégrale :

L'action intégrale agit proportionnellement à la surface de l'écart entre la consigne et la mesure, et elle poursuit son action tant que cet écart n'est pas nul. On dit que l'action Intégrale donne la précision statique (*Elle annule l'erreur statique*). L'action intégrale EST conditionnée par le temps d'intégrale $\boldsymbol{Ti}$.

$$\boldsymbol{\mathcal{U}(t) = \frac{1}{T_i}\int_0^t \mathcal{E}(\tau).d_\tau} \qquad \textbf{(II.12)}$$

Comme dans le cas de l'action proportionnelle, un dosage trop important de l'action intégrale Engendre une instabilité de la boucle de régulation. Pour son réglage, il faut là aussi trouver un compromis entre la stabilité et la rapidité. L'ajout du terme intégral permet d'améliorer la précision mais en contrepartie, il introduit malheureusement un déphasage de (- π : ce qui risque de rendre le système instable du fait de la diminution de la marge de phase. Enfin, le correcteur intégral présente le défaut de saturer facilement si l'écart ne s'annule pas rapidement, ce qui est le cas des systèmes lents. En effet, tout actionneur est limité : un moteur est limité en vitesse, une vanne ne peut pas être plus que totalement ouverte ou totalement fermée. Il se peut que la variable de commande amène l'actionneur à sa limite ce qui suspend la boucle de retour et le système aura une configuration assimilable à une boucle ouverte puisque l'actionneur demeurera saturé indépendamment de la sortie du système. [10]

Quand l'erreur est réduite (*action intégrale non saturée*), il se peut qu'il faille un temps important pour que les valeurs des variables ne soient correctes de nouveau : on appelle

ce phénomène l'emballement du terme intégral. Pour l'éviter, on peut par exemple :

- soit suspendre l'action intégrale quand la commande est saturée ;
- soit appliquer une méthode d'anti-saturation, qui consiste à recalculer le terme intégral pour ne pas saturer.

Pour le cas discret, le terme intégral peut être remplace par la somme des écarts et la différentielle ***dt par Te*** ce qui nous donne le résultat suivant :

$$\boldsymbol{\mathcal{U}(n) = \frac{T_E}{T_I}\sum_{k=0}^{n}\mathcal{E}(k) = \mathcal{U}(n-1) + \frac{T_e}{T_i}\mathcal{E}(n)} \qquad \textbf{(II.13)}$$

$\boldsymbol{Ti}$: Constante de temps intégrale et s'exprime en (s).

Le problème est que lorsque $\mathcal{E}$ redevient nul, le signal U peut avoir atteint une valeur trop élevée pour qu'il y ait équilibre. Autrement, elle permet d'éliminer l'erreur résiduelle en régime permanent.

En résumé, nous affirmons que l'action intégrale est utilisée lorsqu'on désire avoir en régime permanent, une précision parfaite, en outre, elle permet de filtrer la variable à régler d'où l'utilité pour le réglage des variables bruitées telles que la commande.

II.3 .régulation analogique

II.3.1.la transformation de Laplace :

En mathématiques, la transformation de Laplace est une transformation intégrale, c'est-à-dire une opération associant à une fonction f (à valeur dans ou dans) une nouvelle fonction dite transformée de Laplace de f, notée traditionnellement F, via une intégrale.

II.3.2.Remarque :

On note traditionnellement t le paramètre générique de f (formant ainsi $f(t)$), tandis que l'on note plutôt p celui de sa transformée F (on écrit donc F(p)). [10]

La transformation de Laplace est injective et par calcul (ou par usage de tables) il est possible d'inverser la transformation. Le grand avantage de la transformation de Laplace est que la plupart des opérations courantes sur la fonction originale $f(t)$, telle que la dérivation, ou une translation sur la variable t, ont une traduction (plus) simple sur la transformée F(p). Ainsi :

- La transformée de Laplace de la dérivée $f'(t)$ est simplement $pF(p) - f(0^-)$;
- La transformée de la fonction $f(t-\tau)$ (translation) est simplement $e^{-p\tau}F(p)$.

Cette transformation fut introduite pour la première fois sous une forme proche de celle utilisée par Laplace en 1774, dans le cadre de la théorie des probabilités.

la transformée de Laplace mono latérale d'une fonction f (éventuellement généralisée, telle que la « fonction de Dirac ») d'une variable réelle t, à support positif, est la fonction F de la variable complexe p, définie par :

$$\boldsymbol{F(p) = \mathcal{L}\{f(t)\} = \int_{0^-}^{+\infty} e^{-pt} f(t) dt} \qquad \textbf{(II.14)}$$

Plus précisément, cette formule est valide lorsque Ré(p) > α, où α est l'abscisse de convergence (définie plus bas), $-\infty \leq \alpha \leq +\infty$ et f est une fonction localement intégrable à support positif, c'est-à-dire nulle en-dehors de l'intervalle $I = [0, +\infty[$, ou plus généralement un « germe » de distributions définies dans un voisinage ouvert (et borné inférieurement) de l'intervalle $I = [0, +\infty[$ dont la restriction au complémentaire de I dans ce voisinage est une fonction indéfiniment dérivable (voir l'article Transformation bilatérale de Laplace)[1]

Par exemple, lors de l'étude d'une machine à courant continu :

$$\boldsymbol{e(t) = R.i(t) + L\frac{di(t)}{dt}} \qquad \textbf{(II.15)}$$

Dans le domaine fréquentiel devient :

$$\boldsymbol{E(p) = R.I(p) + P.L.I(p)} \qquad \textbf{(II.16)}$$

Dans le domaine de Laplace. Ceci n'est valable qu'à conditions initiales nulles : $i(0) = 0$.

On a utilisé ici des propriétés de la transformation de Laplace, explicitées ci-dessous.

II.3.3.Remarque :

La notation « s » (variable de Laplace) est souvent utilisée dans les pays anglo-saxons alors que la notation « p » est utilisée notamment en France et en Allemagne.

On définit aussi, dans les mêmes conditions que ci-dessus, la transformation de Laplace-Carson par[2] :

$$\boldsymbol{\emptyset(p) = p\int_{0^-}^{+\infty} e^{-pt} f(t) dt} \qquad \textbf{(II.17)}$$

Qui permet d'associer à toute fonction d'une variable $t \to f(t)$une fonction image

$. p \to \emptyset(p)$.

Cette transformation est utilisée par certains ingénieurs car :

- une constante sur [0, +∞ [a pour image la même constante ;
- elle offre dans certains cas une plus grande facilité d'emploi en calcul matriciel et tensoriel.

L'inversion de la transformation de Laplace s'effectue par le biais d'une intégrale dans le plan complexe. À l'aide du théorème des résidus, on démontre la formule de Bromwich **(en)**-Mellin :

$$f(t) = \mathcal{L}^{-1}\{F(p)\} = \frac{1}{2\pi i}\int_{\gamma-i.\infty}^{\gamma+i.\infty} e^{pt}\, F(p)dp \qquad \textbf{(II.18)}$$

Où γ est choisi de sorte que :

- l'intégrale soit convergente, ce qui implique que γ soit supérieur à la partie réelle de toute singularité de F(p).
- et qu'à l'infini, $\lceil F(p) \rceil$tende vers 0 au moins aussi rapidement que.
- $\frac{1}{|p^2|}$

Lorsque cette dernière condition n'est pas satisfaite, la formule ci-dessus est encore utilisable s'il existe un entier *n* tel que $|p^{-n}\ F(p)\ |$ tende vers 0 aussi rapidement que $\frac{1}{|p^2|}$

C'est-à-dire :

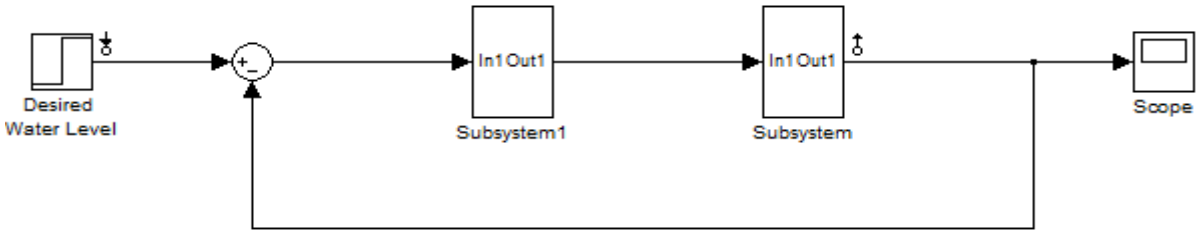
Pour |p| tendant vers l'infini, |F(p)| est majorée par un polynôme en |p|.

En remplaçant F(p) par p^{-n}F(p) dans l'intégrale ci-dessus, on trouve dans le membre de gauche de l'égalité une fonction généralisée à support positif dont la dérivée d'ordre *n* (au sens des distributions) est la fonction généralisée (elle aussi à support positif) cherchée.

En pratique néanmoins, la formule de Bromwich-Mellin est peu utilisée, et on calcule les inverses des transformées de Laplace à partir des tables de transformées de Laplace. [10]

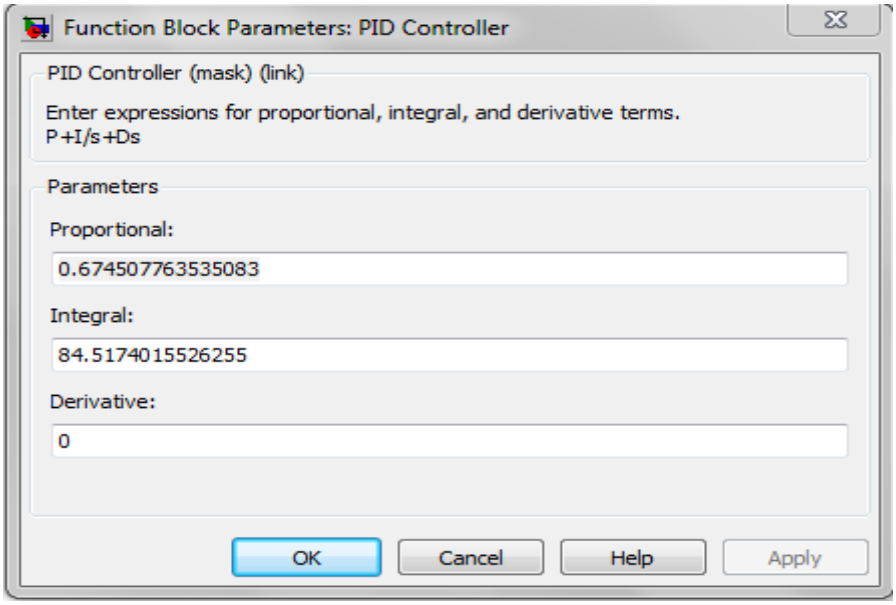
II.4. La conception d'un Compensateur PI Utilisation du Tubing Algorithme Ziegler-Nicholas

Dans cette partie du tutoriel, vous concevez un compensateur en utilisant le PID Ziegler-Nicholas en boucle ouverte algorithme de réglage automatique. Cette méthode de réglage calcule les gains proportionnels, intégraux et dérivés à l'aide du Chien-Hrones-Resnick (CHR) de réglage avec un dépassement de 20%.



FigureII.3.*Schéma bloc du modèle watertank_comp_design*

1eme essaie : pour KI=0.67450 et KP= 84.5174. On obtient la résultat sur la figure ci-dessus



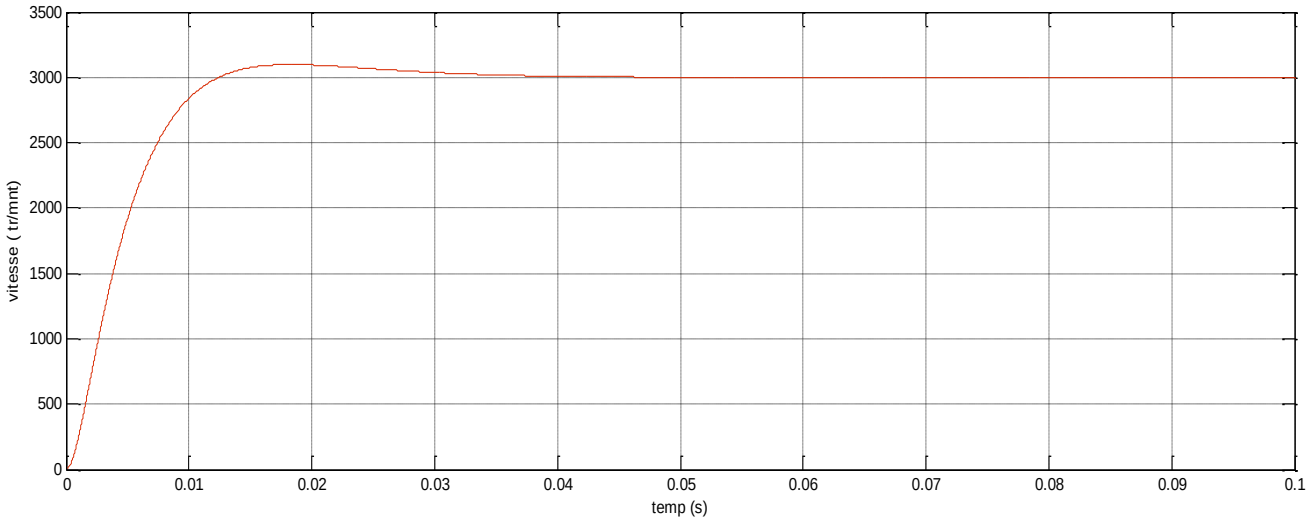
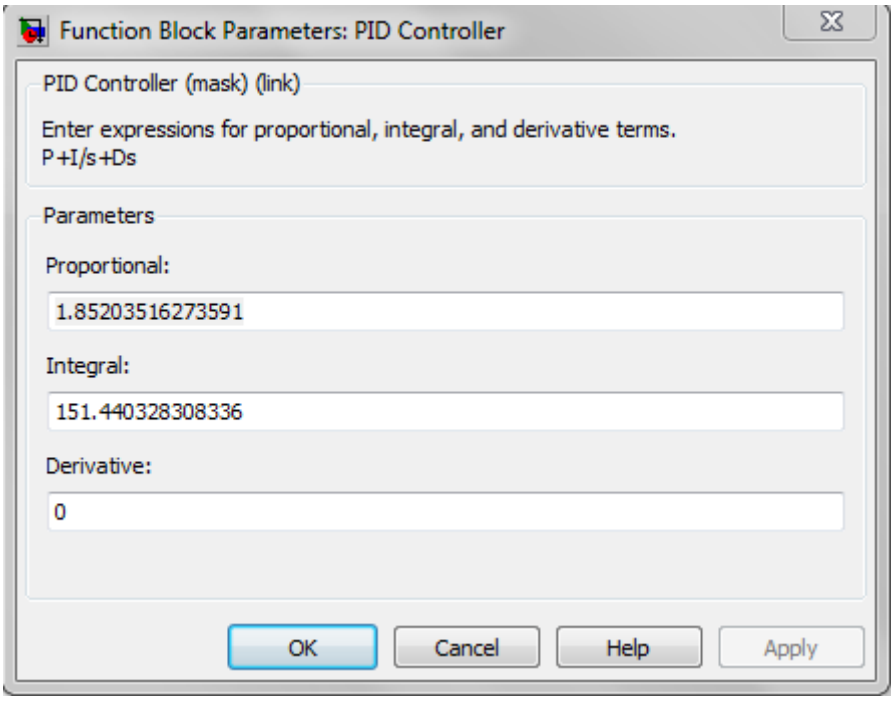


Figure. II.4. *la vitessede 3000 tr/min en fonction de temps.*

2eme essaie : pour KI=1.8520 et KP= 151.4402. On obtient le résultat sur la figure ci-dessus



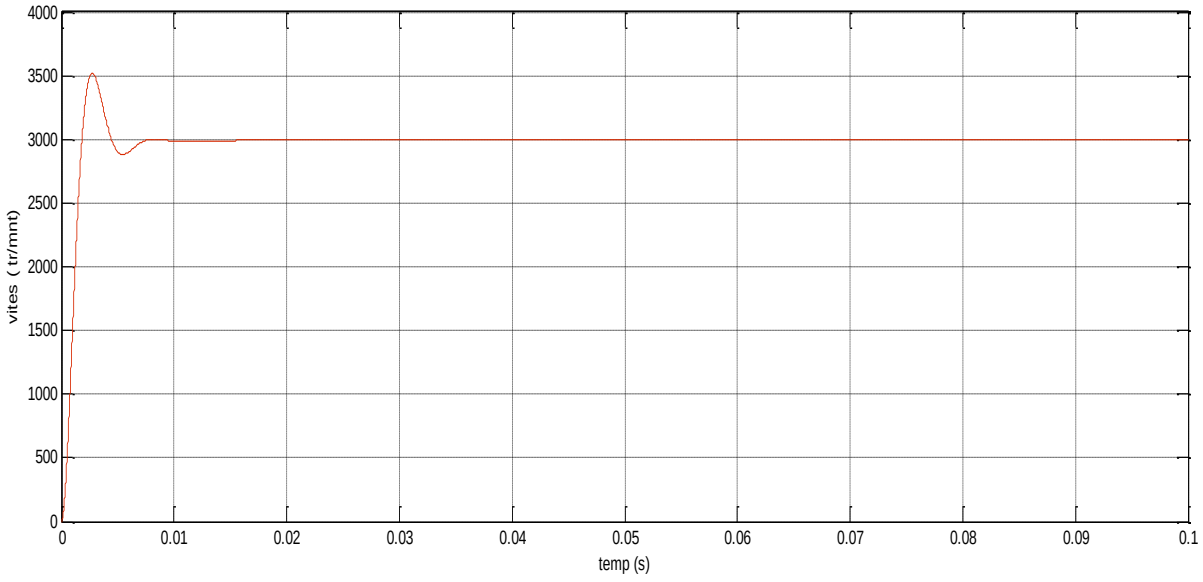
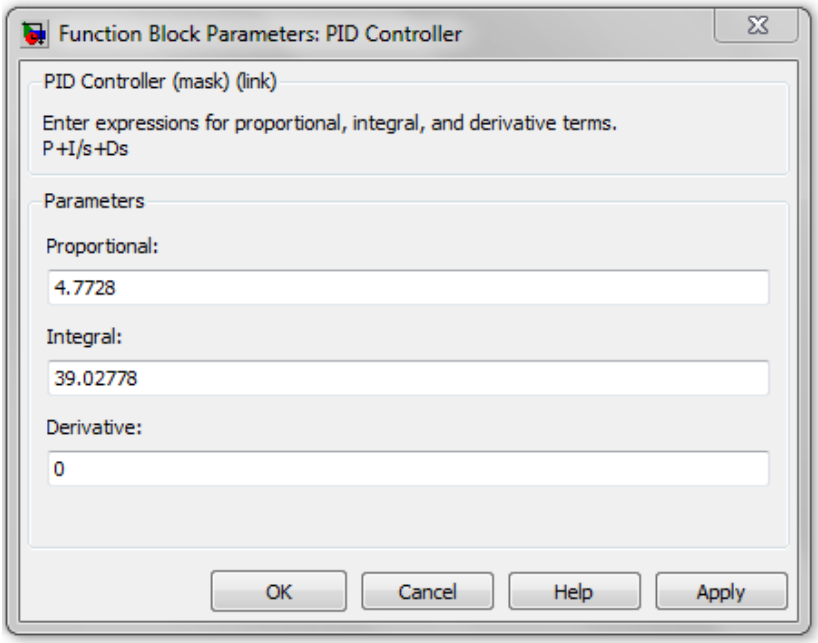


Figure. II.5. *la vitesse de 3000 tr /min en fonction de temps.*

3eme essaie : pour KI=4.7728 et KP= 39.02778. On obtient le résultat sur la figure ci-dessus



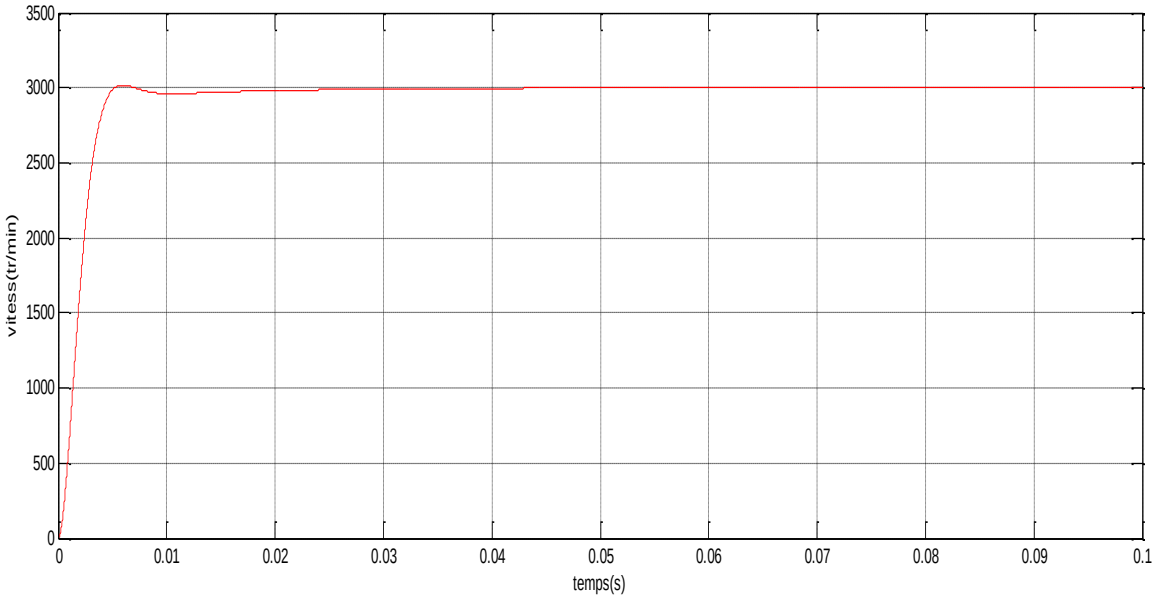


Figure. II.6. *la vitesse de 3000 tr/min en fonction de temps.*

II.5. Régulation numérique :

II.5.1 .*fonction de transfert discrète du PI*

Le schéma bloc de la figure II.7illustre un compensateur PI discret. En effet, la sortie m (kT) du compensateur est la somme d'un gain proportionnel Kp multipliée par l'erreur entre la référence r (kT) et la mesure cm (kT) ainsi que l'intégrale de cette erreur multipliée par le gain intégrale Ki. La sortie de compensateur est donc donnée par :

La fonction du transfert du régulateur numérique PI est donné par :

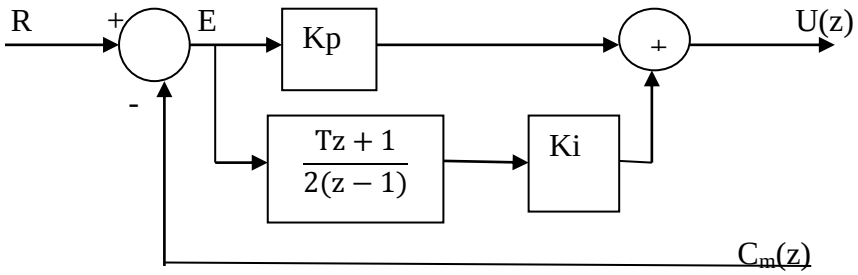


Figure II.7. *Schéma bloc de la fonction de transfert PI. [10]*

$$\frac{U(z)}{E(z)} = Kp + Ki\ \frac{T}{2}\frac{z+1}{z-1} = \frac{Kp(z-1)+Ki\frac{T}{2}(z+1)}{z-1} \qquad \textbf{(II.21)}$$

$$\frac{U(z)}{E(z)} = \frac{Q0\ z+Q1}{z-1} \quad \text{où} \quad \begin{cases} Q0 = Kp + Ki\frac{T}{2} \\ Q1 = -Kp + \frac{T}{2} \end{cases}$$

(II.22)

Avec : E : erreur

U : la commande

De cette fonction de transfert nous avons mis en œuvre la loi de commande (équation récurrente) comme suite :

$z\ U(z) - U(z) = Q0\ z\ E(z) + Q1\ E(z) * z^{-1}$ **(II.23)**

$U(z) - z^{-1}\ U(z) = Q0\ E(z) + Q1\ z^{-1}\ E(z)$

$U(k) - U(k-1) = Q0\ e(k) + Q1\ e(k-1)$

$U(k) = U(k-1) + Q0\ e(k) + Q1\ e(k-1)$

Cela signifie que la commande actuelle U(k) est égale à la somme de la commande précédente plus un coefficient Q0 multiplie par l'erreur actuelle plus un coefficient Q1 multiplie par l'erreur précédente donc notre traitement se base sur deux équations :

$$U(k) = U(k-1) + Q0\ e(k) + Q1\ e(k-1)$$

(II.24)

$$\boldsymbol{m(KT) = K_p\ e(KT) + K_i\ ei(KT)}$$

(II.25)

ou e *(kT)* l'erreur et e_i*(kT)* est l'intégrale de l'erreur qui peut être séparée sur deux intervalles de la façon suivante :

$$\boldsymbol{ei(kT))=\int_0^{KT} e(t)dt=\int_0^{(k-1)T} e(t)dt+\int_{(k-1)T}^{kT} e(t)dt}$$

(II.26)

Selon la définition de e_i *(kT),* la relation II.2 peut être réécrite de la façon suivante :

$$\boldsymbol{ei(KT) = ei\big((k-1)T\big) + \int_{(k-1)T}^{KT} e(t)dt} \qquad \textbf{(II.27)}$$

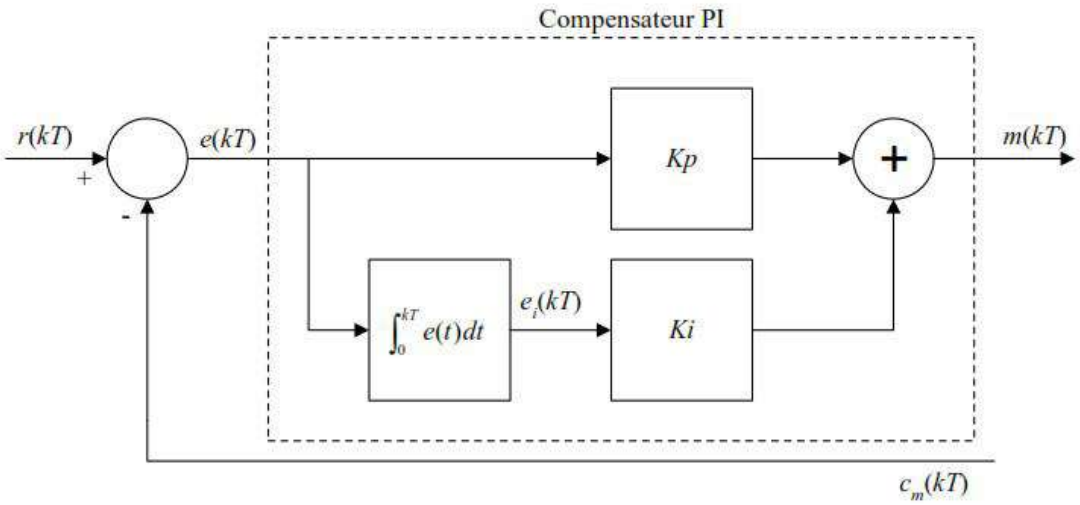


Figure II.8. *Schéma blocs du compensateur PI.* [10]

Le problème est alors d'obtenir l'intégrale de e(t) à partir du signal discret e(kT) qui est connu seulement aux temps *kT* pour k=1,2,3,….(aux périodes d'échantillonnages). En fait, l'intégrale de e(t) ne peut être obtenue de façon exacte ; elle ne peut qu'être approximée. Les trois approximations les plus populaires de cette intégrale sont l'approximation rectangulaire, l'approximation rectangulaire devancée et l'approximation trapézoïdale. [11]

II.5.2.approximation rectangulaire

Telle que décrite par la figure II.2, l'approximation rectangulaire de l'intégrale est donnée par la relation suivante :

$$\boldsymbol{\int_{(k-1)T}^{kT} e(t)dt \sim Te\ ((K-1)\ T)} \qquad \textbf{(II.28)}$$

Ainsi, en remplaçant la relation (II.4) dans la relation, on obtient la relation entre l'entrée et la sortie du système :

$$\boldsymbol{ei(KT) = ei((k-1)T + Te((k-1)T)} \qquad \textbf{(II.29)}$$

En appliquant la transformée en Z de part et d'autre de la relation tout en considérant des conditions initiales nulles, la propriété du retard nous permet d'obtenir la relation suivante :

$$\boldsymbol{E_i(z) = z^{-1}E_i\,(Z) + Tz^{-1}E(z)} \qquad \textbf{(II.30)}$$

La fonction de transfert entre la sortie du compensateur et l'erreur est donc donnée par relation suivante :

$$\frac{\boldsymbol{Ei(z)}}{\boldsymbol{E(z)}} = \frac{\boldsymbol{T}}{\boldsymbol{z-1}} \qquad \textbf{(II.31igure}$$

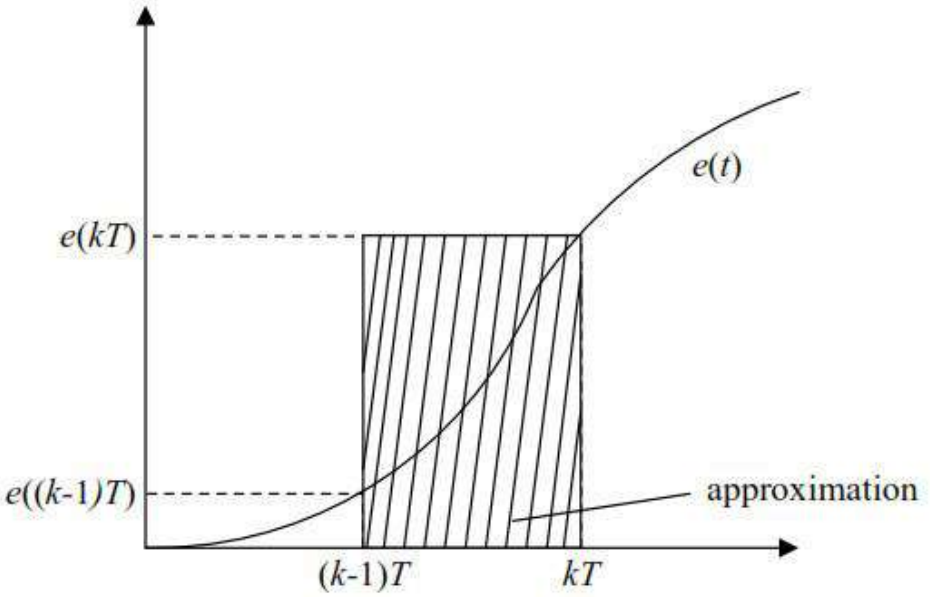


Figure II.9. *Approximation rectangulaire devancée.* [10]

II.5.3. approximation rectangulaire devancée

Telle que décrite par la figure II.3, l'approximation rectangulaire devancée de l'intégrale est donnée par la relation suivante

(II.32)

Ainsi, en remplaçant la relation (II.8) dans la relation (II.3), on obtient la relation entre l'entrée et la sortie du système :

$$\mathrm{ei(kT) = ei((k-1)T) + Te(kT)} \qquad \text{(II.33)}$$

En appliquant la transformée en Z de part et l'autre de la relation tout en considérant des conditions initiales nulles, la propriété du retard nous permet d'obtenir la relation suivante :

$$\mathrm{Ei(z) = Ei(z) + TE(z)} \qquad \text{(II.34)}$$

La fonction de transfert entre la sortie du compensateur et l'erreur est donc donnée par la relation suivante :

$$\frac{Ei(z)}{E(z)} = \frac{Tz}{z-1} \qquad \text{(II.35)}$$

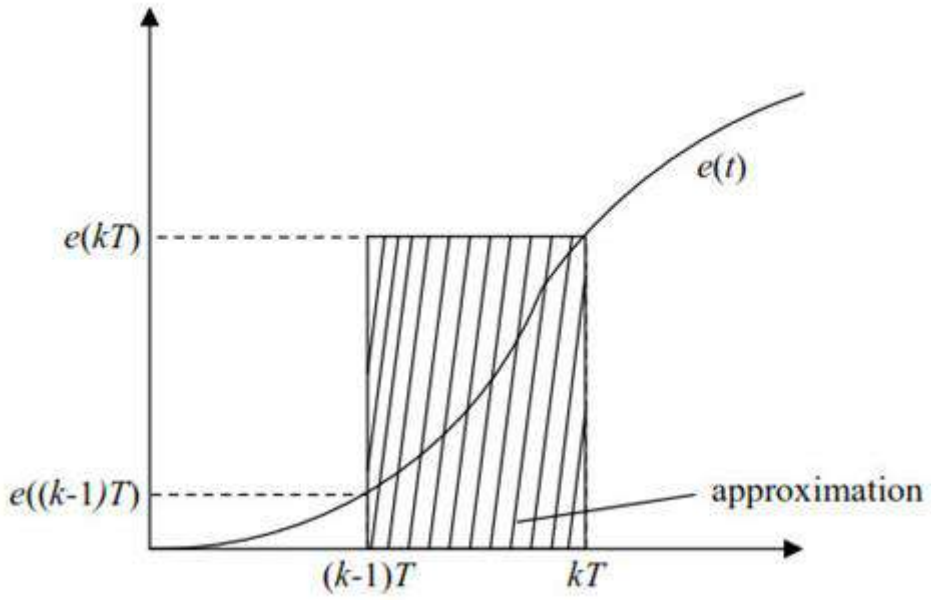


Figure II.10. App*roximation rectangulaire devancée. [10]*

II.5.4. approximation trapézoïdale

Telle que décrite par la figure II.4, l'approximation trapézoïdale de l'intégrale est donnée par la relation suivante :

$$(e((k-1)T) + e(kT)) \qquad \text{(II.36)}$$

Ainsi, en remplaçant la relation (II.12) dans la relation(II.3), on obtient la relation entre l'entrée et la sortie du système :

$$\mathrm{ei(kT)} = \mathrm{ei((k-1)T)} + \ (\mathrm{e((k-1)T)} + \mathrm{e(kT)}) \qquad \text{(II.37)}$$

En appliquant la transformée en z de part et l'autre de la relation tout en considérant des conditions initiales nulles, la propriété du retard nous permet d'obtenir la relation suivante :

$$\mathrm{Ei(z)} = \mathrm{Ei(z)} + \ (\mathrm{E(z)} + \mathrm{E(z)}) \qquad \text{(II.38)}$$

La fonction de transfert entre la sortie du compensateur et l'erreur est donc donnée par la relation suivante :

$$\frac{Ei(z)}{E(z)} = \frac{Tz+1}{2\,z-1} \qquad \text{(II.39)}$$

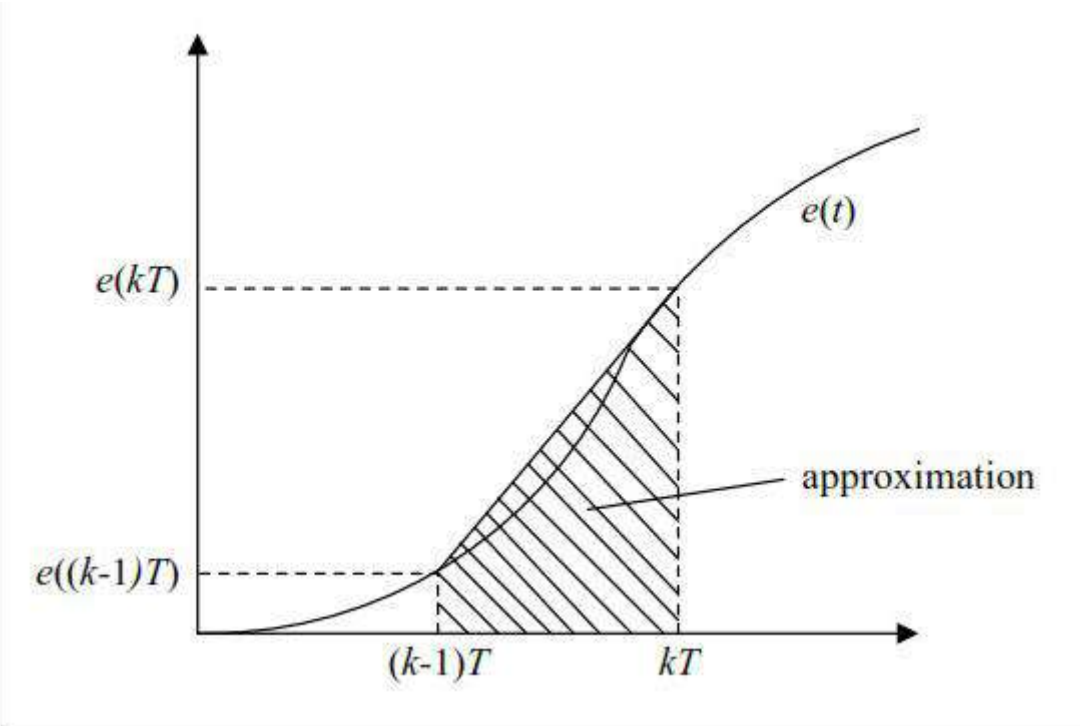


Figure II.11. *Approximation trapézoïdale. [10]*

II.5.3.. fonction de transfert du PI avec l'approximation trapézoïdale

Maintenant que nous avons trouvé les fonctions de transfert qui approxime l'intégrale dans le domaine discret, il suffit de remplacer une de ces approximations dans le schéma bloc du compensateur PI de la figure II.1. Si on considère l'approximation trapézoïdale, qui est visiblement la plus précise, le schéma bloc du compensateur PI dans le domaine de z devient celui de la figure II.5. [11]

La fonction de transfert du compensateur PI avec une approximation trapézoïdale de l'intégrale est donc donnée par la relation suivante :

$$\frac{M(z)}{E(z)} = Kp + Ki\frac{T}{2}\frac{z+1}{z-1} \qquad \textbf{(II.40)}$$

En plaçant les deux termes sous un même dénominateur, on obtient

$$\frac{M(z)}{E(z)} = \frac{Q0Z+Q1}{z-1} \qquad \textbf{(II.41)}$$

Où

$$\mathbf{Q0 = Kp + Ki\,\frac{T}{2}} \qquad \mathbf{Q1 = -Kp + Ki\,\frac{T}{2}} \qquad \textbf{(II.42)}$$

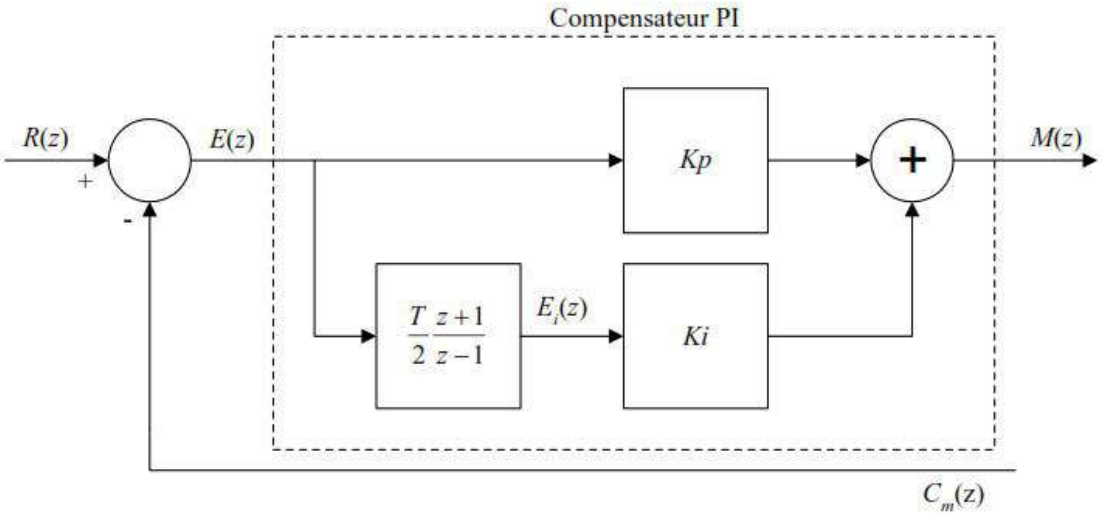


Figure II.12. *Schéma bloc en z du compensateur PI.* [10]

II.6.LA PARTIE DE SIMULATION

Dans ce chapitre on a modélisé la machine à courant continu a travers d'un régulateur proportionnel intégral sous MATLAB /SIMULINK.

On a pour la machine à courant continu :

• Equations électriques :

Prenant en compte la résistance Ra et l'inductance La du circuit d'induit, du collecteur, des balais et des connexions, et en les supposant toutes deux constantes (pas de variation due à l'échauffement ni à la saturation magnétique), l'équation de tension induite

S'écrit :

$$u_a(t) = \text{Ra.ia}(t) + \frac{d\Psi}{dt} = \text{Ra.ia}(t) + \frac{(d(N.\phi f)}{dt} \qquad \textbf{(II.43)}$$

$$\mathbf{ua(t) = Ra.ia(t) + La\,\frac{dia}{dt} + em(t)} \qquad \textbf{(II.44)}$$

• Equations électromécaniques :

La tension induite em(t), appelée FEM ("force électromotrice" dans l'optique de l'exploitation en générateur) est proportionnelle à la vitesse angulaire w(t) et au flux inducteur Φf(t) :

$$\boldsymbol{em(t) = k.\phi f\,(t).\omega(t)} \qquad \textbf{(II.45)}$$

K est une constante dépendant de la construction de la machine. La première équation montre qu'em (t) s'oppose à Ua (t), c'est à dire que le moteur réagit en créant une FEM

Le couple électromagnétique Tem(t) développé a pour expression :

$$\boldsymbol{Tem(t) = k.\phi f\,(t).ia(t)} \qquad \textbf{(II.46)}$$

• Equation mécanique :

Le moteur en rotation est décrit par l'équation (de la dynamique) d'équilibre suivante :

$$\boldsymbol{Jt\ \frac{d\omega}{dt} = Tem(t) - B\omega(t) - Tr(t) - T(t)} \qquad \textbf{(II.47)}$$

Où

J : inertie totale entraînée ;

B: coefficient de frottement visqueux

Tr : couple résistant

T : couple de frottement de coulomb

Voilà en somme les équations régissant le fonctionnement du moteur. Mais pour mieux les exploiter, nous allons utiliser leurs transformées de Laplace.

Les transformées de Laplace des équations sont les suivantes :

$$\mathbf{ua(t) = Ra.ia(t) + La\frac{dia}{dt} + em(t) \rightarrow ua - Em = Ra.Ia + s.La.I_a} \qquad \textbf{(II.48)}$$

$$\mathbf{em(t) = k.\phi f(t).\omega(t) \rightarrow Em = k.\phi f.\Omega} \qquad \textbf{(II.49)}$$

$$\mathbf{Tem(t) = k.\phi f(t).ia(t) \rightarrow Tem = k.\phi f.Ia} \qquad \textbf{(II.50)}$$

$$\mathbf{Jt\frac{d\omega}{dt} = Tem(t) - Tr(t) - B\omega(t) - T \rightarrow Tem - T - B\omega(t) - T = s.Jt.} \qquad \textbf{(II.51)}$$

Matlab est un logiciel de calcul matriciel à syntaxe simple. Avec ses fonctions spécialisées, Matlab peut être aussi considéré comme un langage de programmation adapté pour les problèmes scientifiques.

Matlab est un interpréteur: les instructions sont interprétées et exécutées ligne par ligne.

Matlab fonctionne dans plusieurs environnements tels que x-windows, windows, macintosh. Il existe deux modes de fonctionnement :

- Mode interactif: Matlab exécute les instructions au fur et à mesure qu'elles sont données par l'usager
- Mode exécutif: Matlab exécute ligne par ligne un "fichier m" (programme en langage Matlab) . [11]

II.6.1. Les schémas blocs

Les transformées de Laplace obtenues nous permettent de modéliser le moteur sous forme de schéma bloc ou schéma fonctionnel. Ces schémas présentent les fonctions de transfert suivant les paramètres d'entrée et de sortie considérées.

A notre travail, on considère la tension aux bornes de l'induit comme une entrée et la vitesse de moteur comme une sortie, nous obtenons le schéma bloc tension – vitesse. Ce dernier nous permet de voir l'évolution de la vitesse avec la tension.

La création du bloc de notre moteur (schéma bloc d'un Motors MCC) avec la jonction de régulateur PI par MATHLAB. Sa donne la figure suivante :

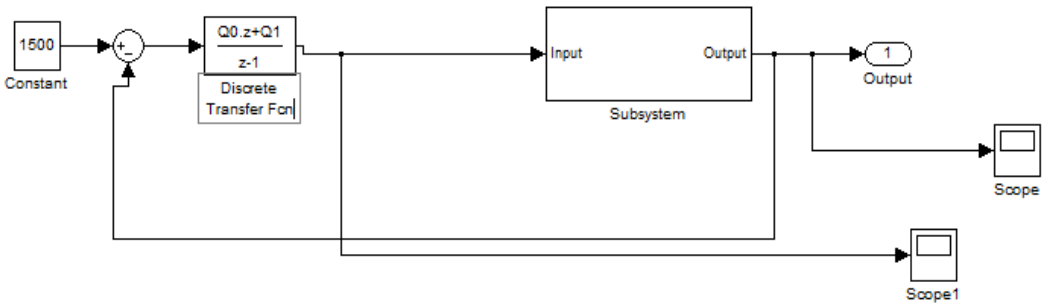


Figure II.13. *Schéma bloc globale de régulation de MCC par PI en Z*

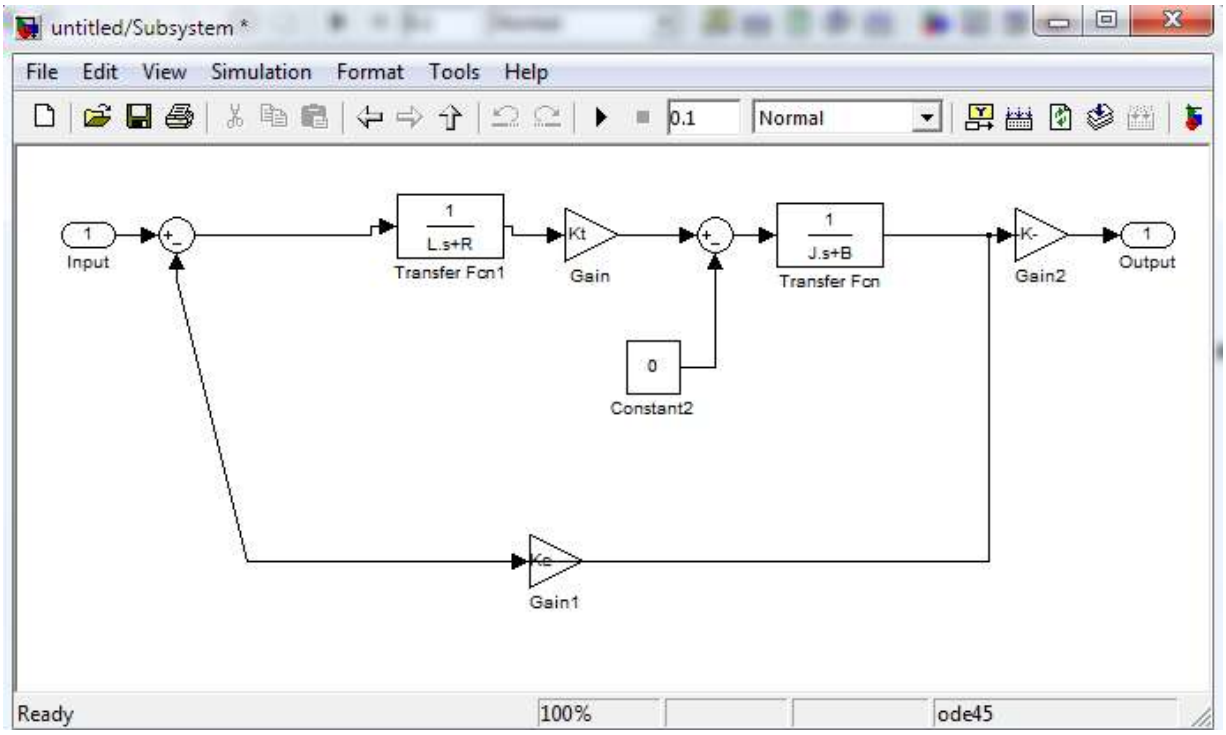


Figure. II.14. *Schéma bloc MCC*

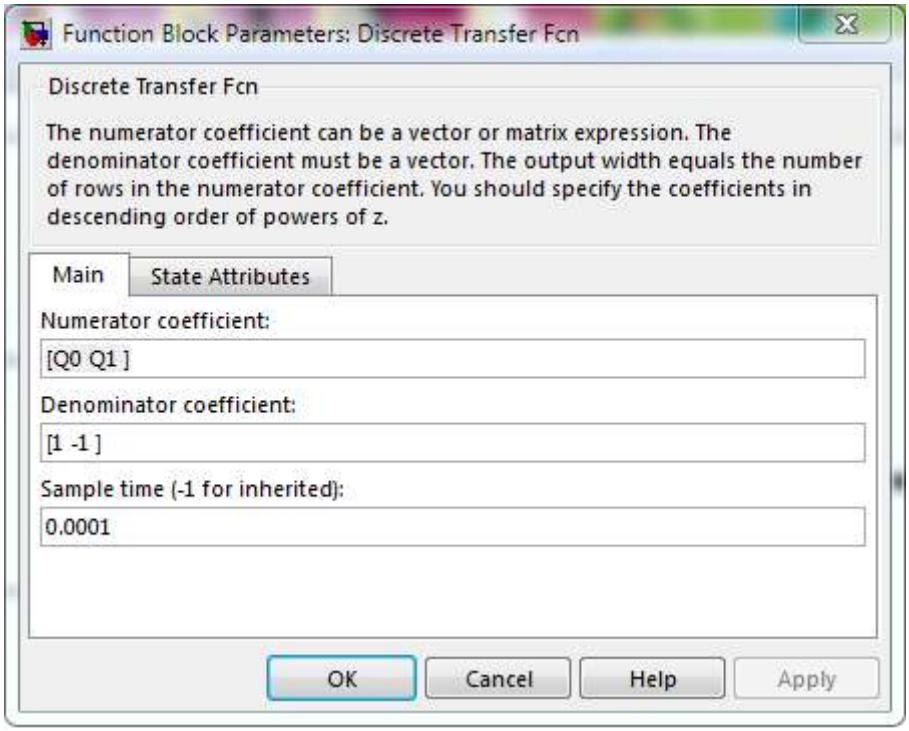


Figure. II.15. *Schéma bloc de la régulateur PI et les gains*

Après l'exaction de la simulation on obtient le résultat de la régulation de la vitesse en fonction de temps

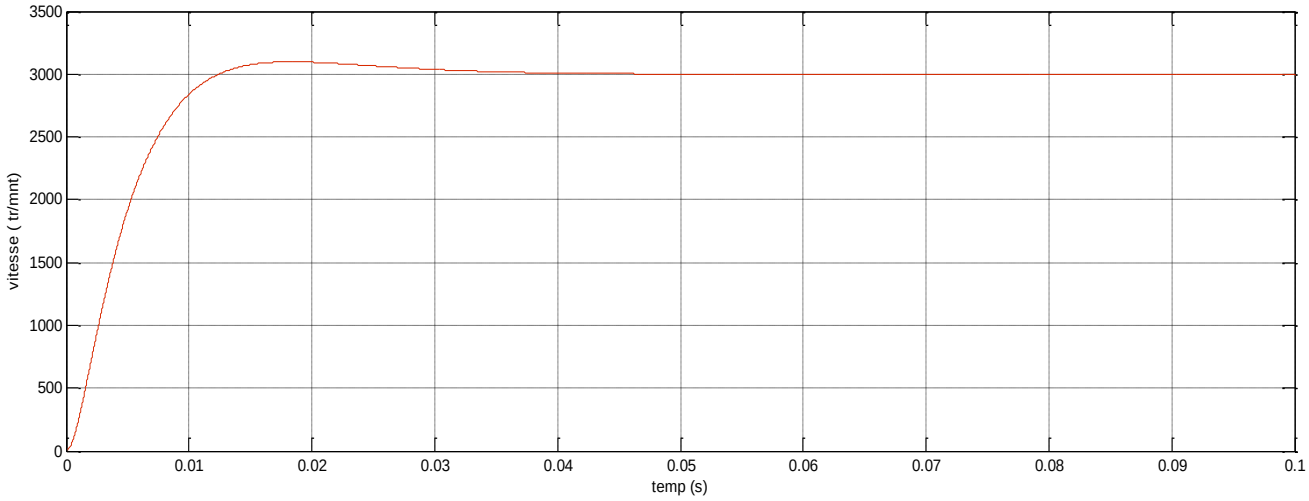


Figure. II.16. *la vitesse de 3000 tr/min en fonction de temps*

2eme essaie : pour KI=4.77 et KP= 390.27. On obtient le résultat sur la figure ci-dessus

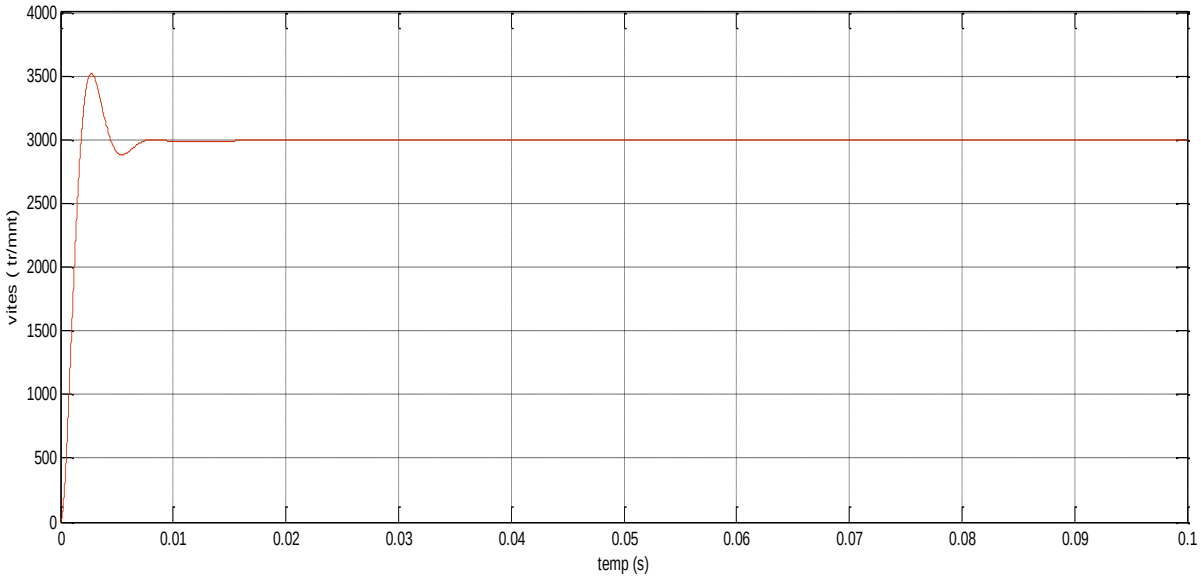


Figure. II.17. *la vitesse en fonction de temps*

2eme essaie : pour KI=4.77 et KP= 39.27. On obtient le résultat sur la figure ci-dessus

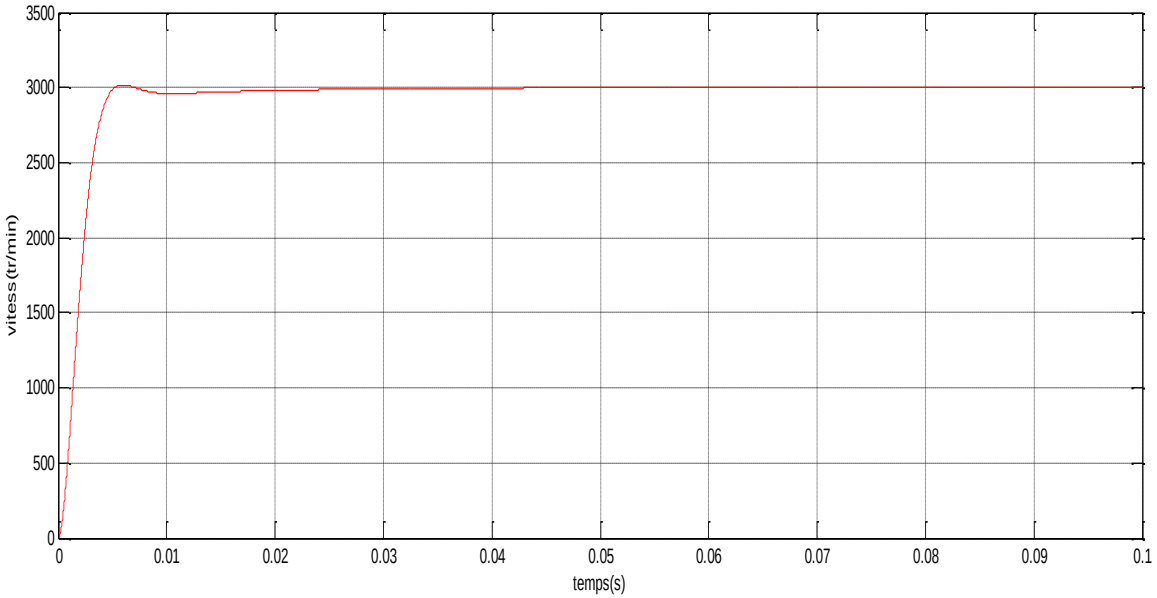


Figure. II.18. *la vitesse de 3000 tr/min en fonction de temps*

II.6.3.*Discussion*

On constate que la vitesse suit bien sa référence imposé (Ω réf= 314rd/s entre l'instant (t=0 jusqu'à t=0.1s), cela confirme les performances obtenue par le régulateur PI.

Les coefficients Ki, Kp sont déterminer par la méthode d'emplacement de pôles

On trouve par la méthode Ziegler nicols les meilleures configurations :

$$kp = 4.77$$
$$ki = 39..27$$

II.7. CONCLUSION

Ce chapitre contient deux parties, dans la première partie on a fait une étude théorique sur le régulateur PI, dans la deuxième partie on a fait une régulation numérique avec le compensateur PI (Nicholas Ziegler méthode) dont l'outil de simulation est le logiciel MATLAB Simulink.

Au la fin on trouve les meilleurs gains pour les implémenter dans notre régulateur PI

Et on obtient des bons résultats.

Chapitre III

Simulation et Realisation

III.1 INTRODUCTION

Le troisième chapitre a fait l'objet de la partie expérimentale de notre projet. En effet, nous avons décrit toutes les étapes de notre réalisation ainsi que les outils utilisés aussi ainsi hardware que software.

Dans ce chapitre on va implémenter le régulateur numérique (PI) dans un microcontrôleur PIC16f877A dont l'outil de programmation est le logiciel mikroCPro et l'outil de test (simulation) est le logiciel ProteusISIS.

Après on va faire les essaies pratiques pour voir les résultats des expérimentes

III .2 LES OUTILS DE DEVELOPPEMENTS

Dans le but de faire commander le moteur à courant continue à travers le pic16F877A .on doit combiner les trois outils suivant :

a _ proteusISIS pour la simulation.



b _ le mikroC pro pour la programmation du pic.

c_ timer calculator pour genére un code de minuiterie pour le resgistre T1CON du pic.

III .2.1. Le proteus ISIS

III .2.1. Introduction à PROTEUS

Proteus est une suite de conception aussi connu comme Virtual Modeling Système (VSM) offrant la possibilité de simuler le code micro-contrôleur et aussi des circuits.

Donc, si vous êtes prêt à concevoir le matériel et le logiciel ceci est un excellent outil pour jouer avec. Dans ce cas, Proteus ISIS est pour la simulation de la forme schématique du matériel ainsi que le code micro-contrôleur. Il est possible de développer et de conceptions de test avant un prototype physique est construit.

Proteus ISIS combine la facilité d'utilisation avec des outils d'édition puissants comme un très haut degré de contrôle sur l'apparence de dessin, en termes de largeurs de lignes, styles de remplissage, les polices, etc .. Il est capable de supporter la capture schématique pour la simulation.

III .2.1.2.Présentation générale de proteus

Le logiciel PROTEUS est composé de trois modules:

- L’éditeur de schéma ISIS
- ISIS produit d’une part, une liste d’équipotentiels qui peut être utilisée par le simulateur LISA et l’outil de conception de circuit imprimé ARES, d’autre part, une liste de matériel et des rapports de contrôle des règles électriques.
- ARES est un module de conception de circuit imprimé compatible Windows, 98 2000 et XP.

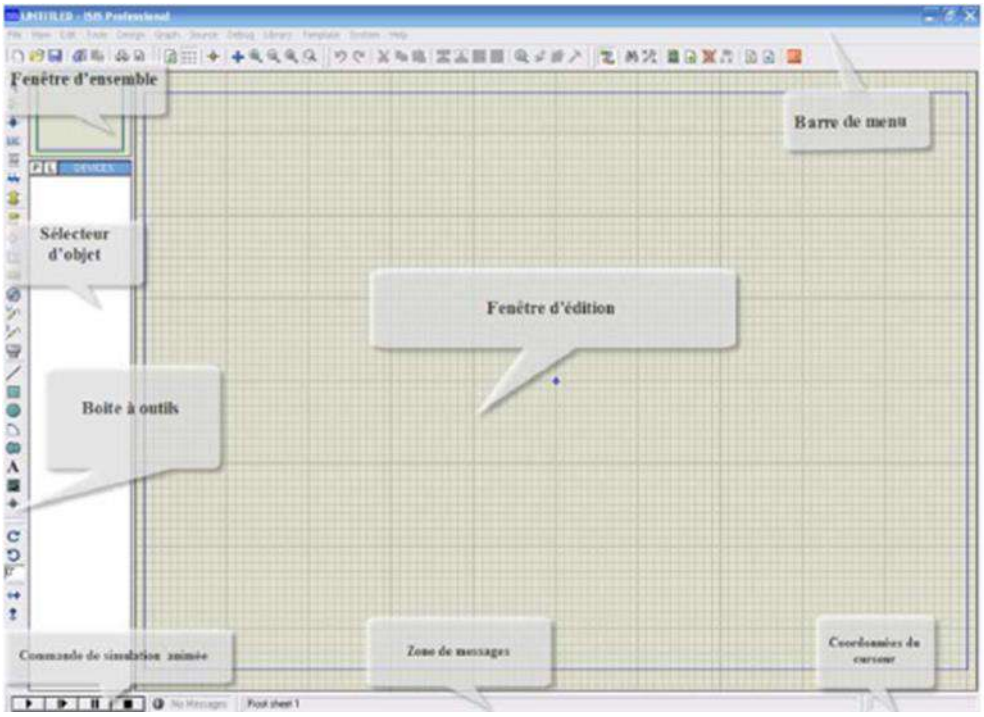
Il permet le placement des composants en mode automatique, manuel ou semi-automatique et le routage des liaisons sur plusieurs couches en mode automatique manuel ou semi-automatique.

III .2.1..3. L’éditeur de schéma ISIS

Isis est un éditeur de schémas qui intègre un simulateur analogique, logique ou mixte.

Toutes les opérations se passent dans cet environnement, aussi bien la configuration des différentes sources que le placement des sondes et le tracé des courbes.

III .2.1.4. Interface utilisateur



FigureIII.1.*Interface utilisateur ISIS.*

III.2.2 .mikroC PRO

III.2.2.1Définitions

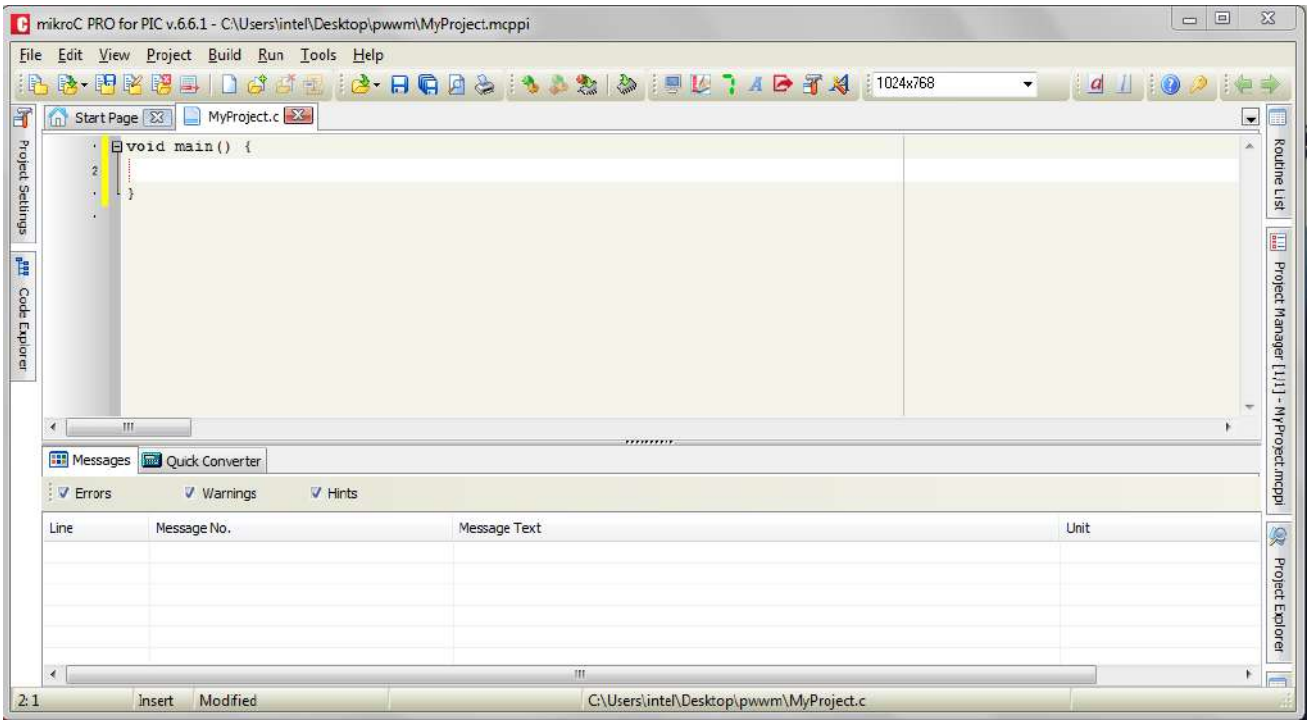
Le MikroC est un puissant outil pour les microcontrôleurs PIC. Il est conçu pour fournir au Programmeur la solution la plus simple possible pour développer des applications pour les systèmes embarqués, sans compromettre les performances ou le contrôle.

Le MikroC permet de développer et déployer des applications complexes:

- Écrire un code source C en utilisant le très avancé éditeur de code.
- Utiliser les bibliothèques du MikroC fait accélérer considérablement le développement (Acquisition de données, la mémoire, affichage, les conversions, les communications.....).
- Surveiller la structure du programme variables et fonctions dans l'explorateur de code.
- Inspecter le déroulement du programme et déboguer la logique d'exécution avec L'intégration debugger, obtenez des rapports détaillés et des graphiques sur les statistiques du Code

Nouvelle version appelée ***mikroC PRO*** dispose de très nombreuses améliorations du compilateur ***mikroC*** : nouvelles variables utilisables, nouvelle interface IDE, amélioration des performances du linker et de l'optimisateur, cycle de compilation plus rapide, code machine généré plus compact (jusqu'à 40 % suivant les cas), nouveaux PIC supportés, environnement de développement encore plus ergonomique, nouveaux exemples d'applications, etc...

- On obtient l'interface finale.



- À la fin de la création de nouveau projet.
- On obtient un fichier .hex (forme hexadécimal).

III.2.3. logiciel Win Pic 800

Dans la phase terminale, une fois le fichier source compilé et simulé, il va falloir transférer le fichier code-machine dans la mémoire du PIC. Pour cela, nous avons utilisé un programmateur (USB auto alimenté) et son logiciel de transfert adapté.

Figure III.6.illustre la fenêtre de configuration du programmateur géré par Winpic800.

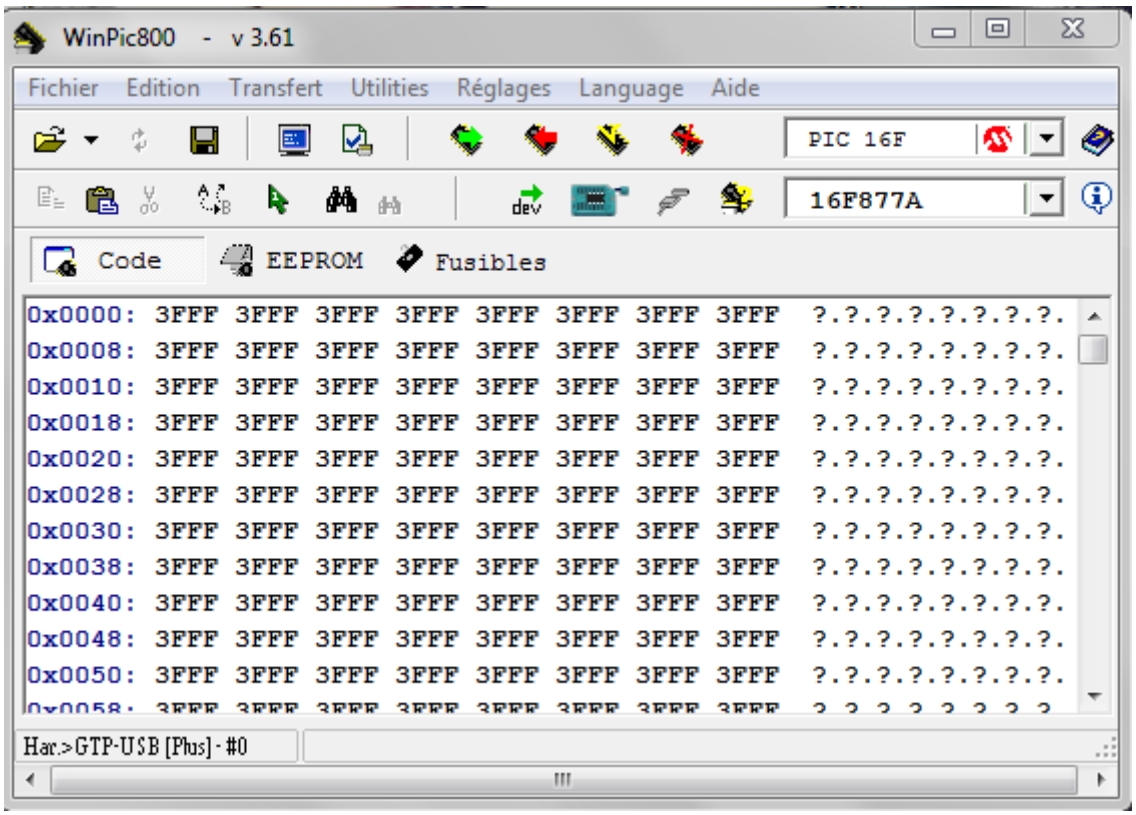


Figure III.2. *Configuration du programmateur géré par Winpic800.*

Pour pouvoir transférer le programme dans la mémoire du PIC on procède comme suit :

- On lance la détection automatique du PIC en cliquant sur l'icône "Détection Device".

La puissance de ce logiciel est capable de programmer une multitude de composants (différent PIC et EEPROM) cependant il est recommandé d'effacer le PIC avant toute nouvelle programmation.

- On charge ensuite le fichier *.HEX de notre programme en cliquant sur l'icône "Ouvrir". Et enfin transfert du fichier *.HEX d'un la mémoire de pic " (voire figureIII.7).

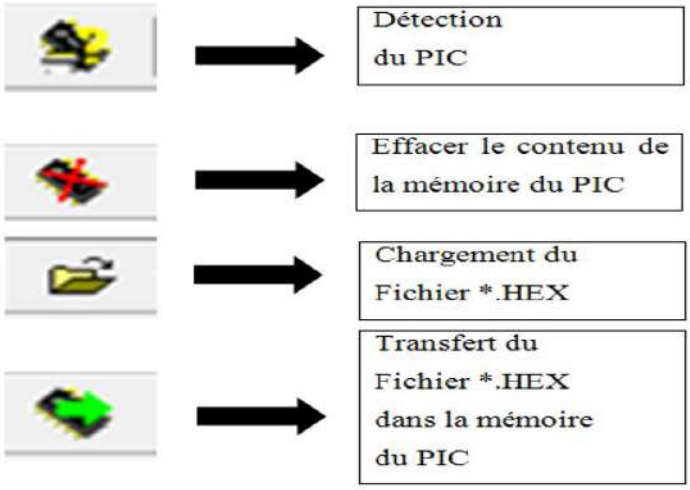
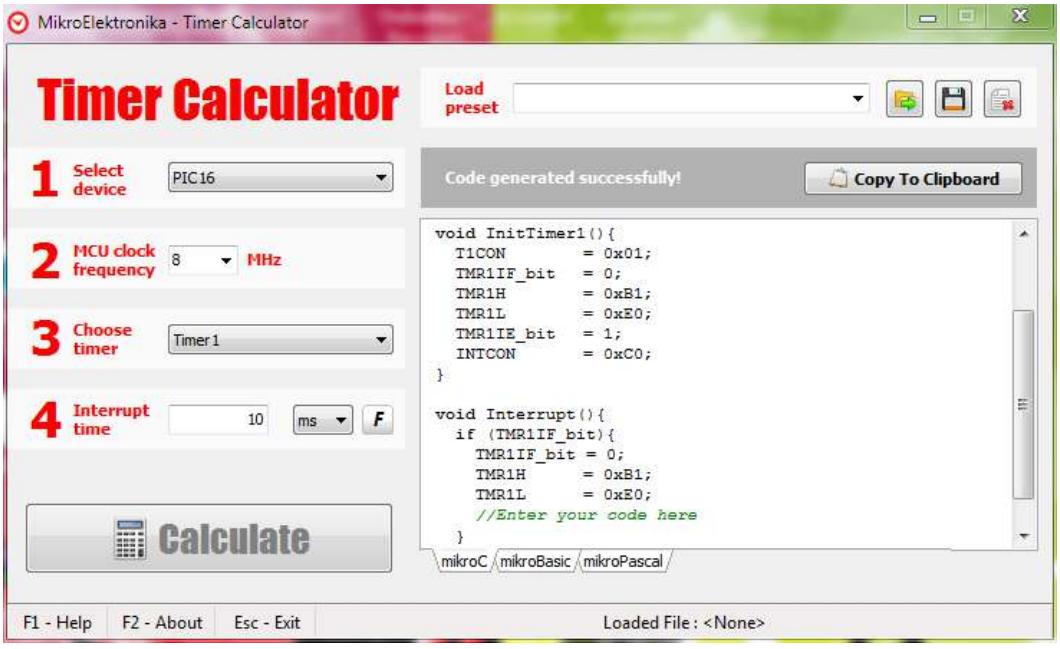


Figure III.3. *Les étapes de transfert d'un programme.*

III.4.timer calculateur

Le mikroElektronika Timer est une application Windows autonome destiné à être utilisé en tant que générateur de code pour le registre T1CON de TIMER1 de notre pic.



III.5. simulation et réalisation

III.5. 1.Introduction

La simulation consiste à faire une étude théorique sur les données pour faire approximé les résultats désirés.

La réalisation consiste a faire l'approximation entre les résultats théoriques et les résultats pratiques.

Dans ce partie on commence par simulie notre commande étape par étape pour approcher a la conception complet de la commande de MCC par un MCC.

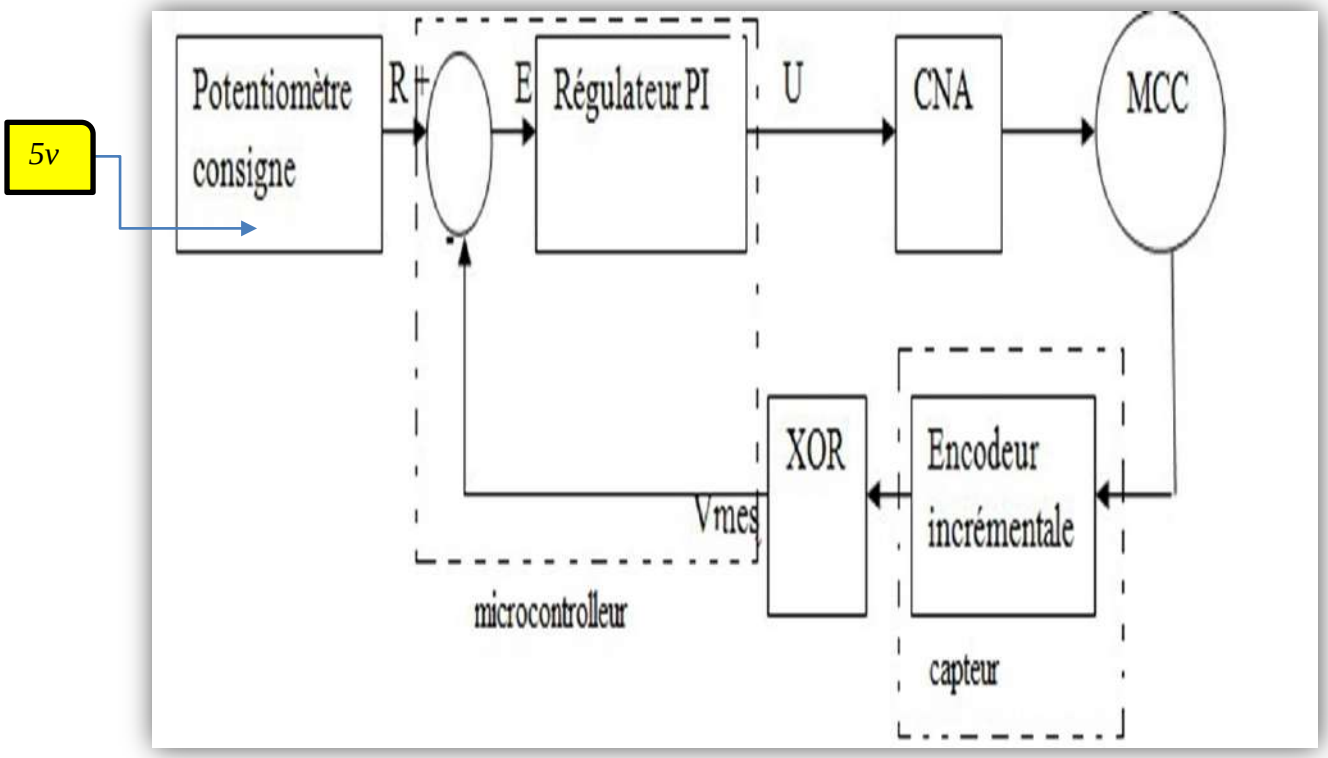


Figure III.4.*Schéma synoptique de la chaine de régulation de la vitesse de MCC.*

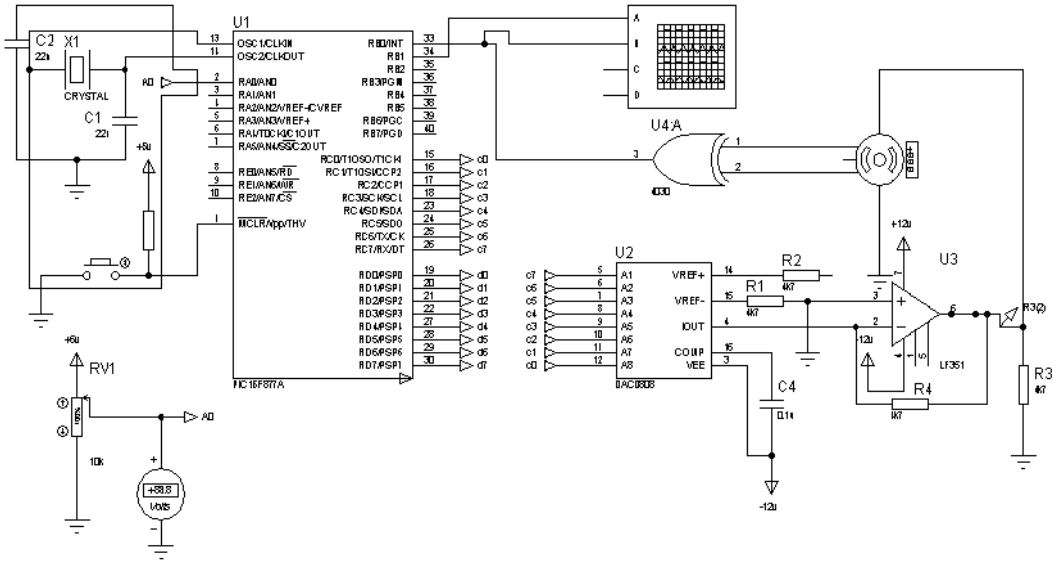


Figure III.5.*Le schéma global sous logiciel proteus configuré par le DAC.*

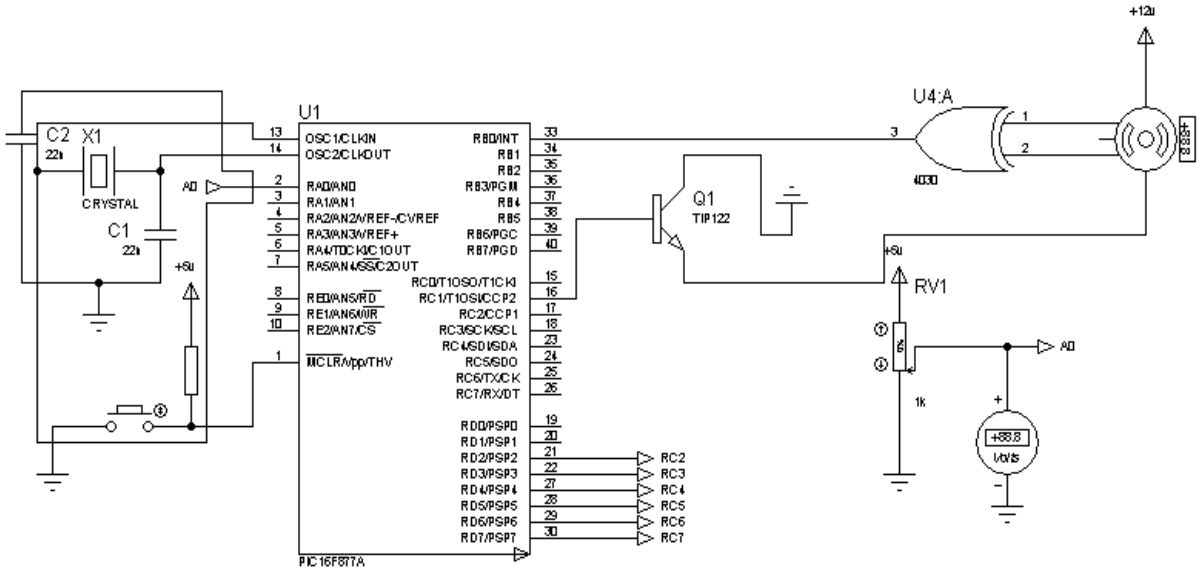


Figure III.6. *Représente la simulation sous proteus configuré par PWM.*

III.5.2.Organigramme

Ce projet avait comme objectif d'établir un programme qui permet de compter le nombre des interruptions générés par RB0 et le temps nécessaire pour ces interruptions par TMR1

On donne un organigramme qui représente la génération de la commande de la vitesse de MCC par les fonctions récurrentes nécessaire pour la régulation :

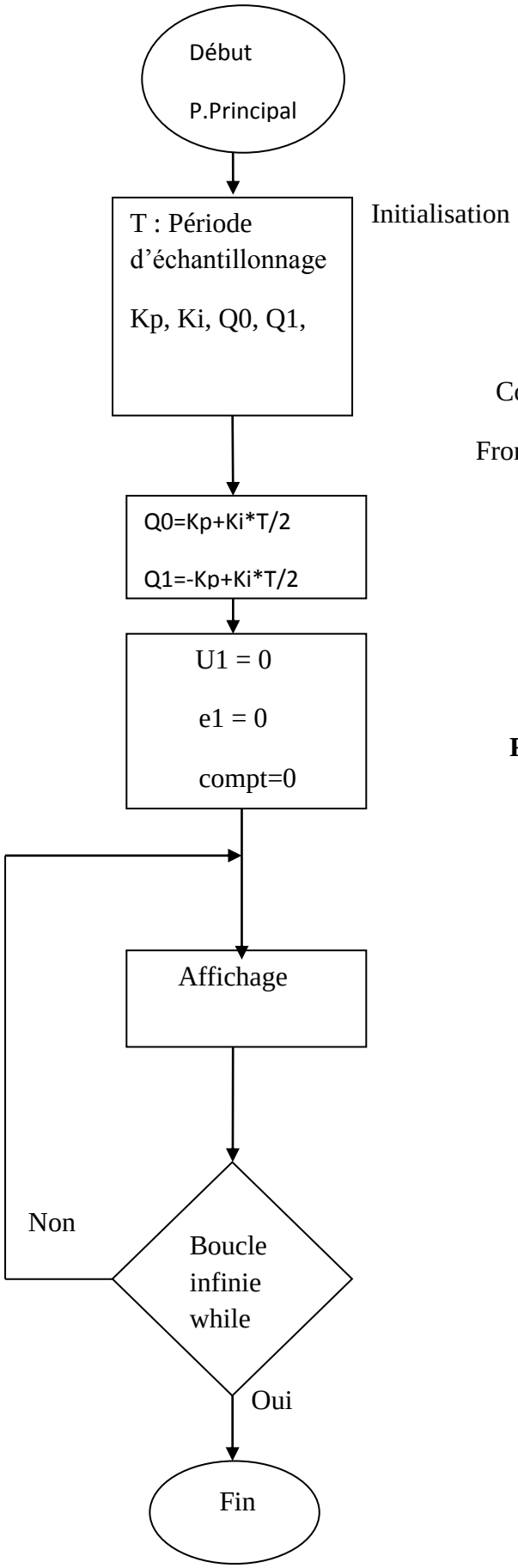


Figure III.7.*organigramme de programme principale.*

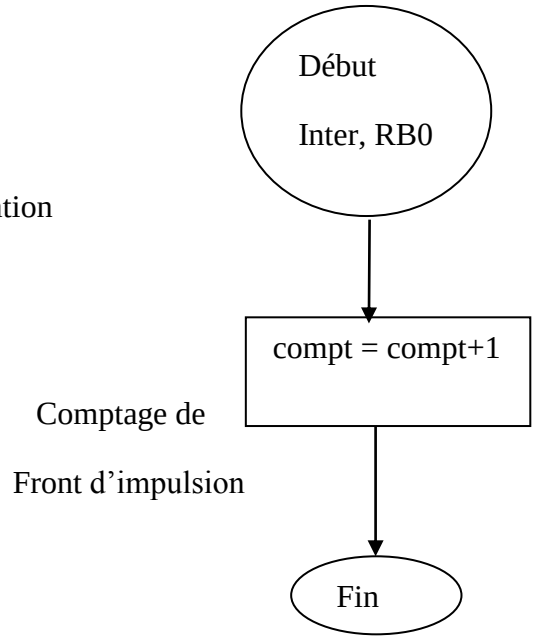


Figure III.8. *Organigramme de Programme De comptage de nombre de tours.*

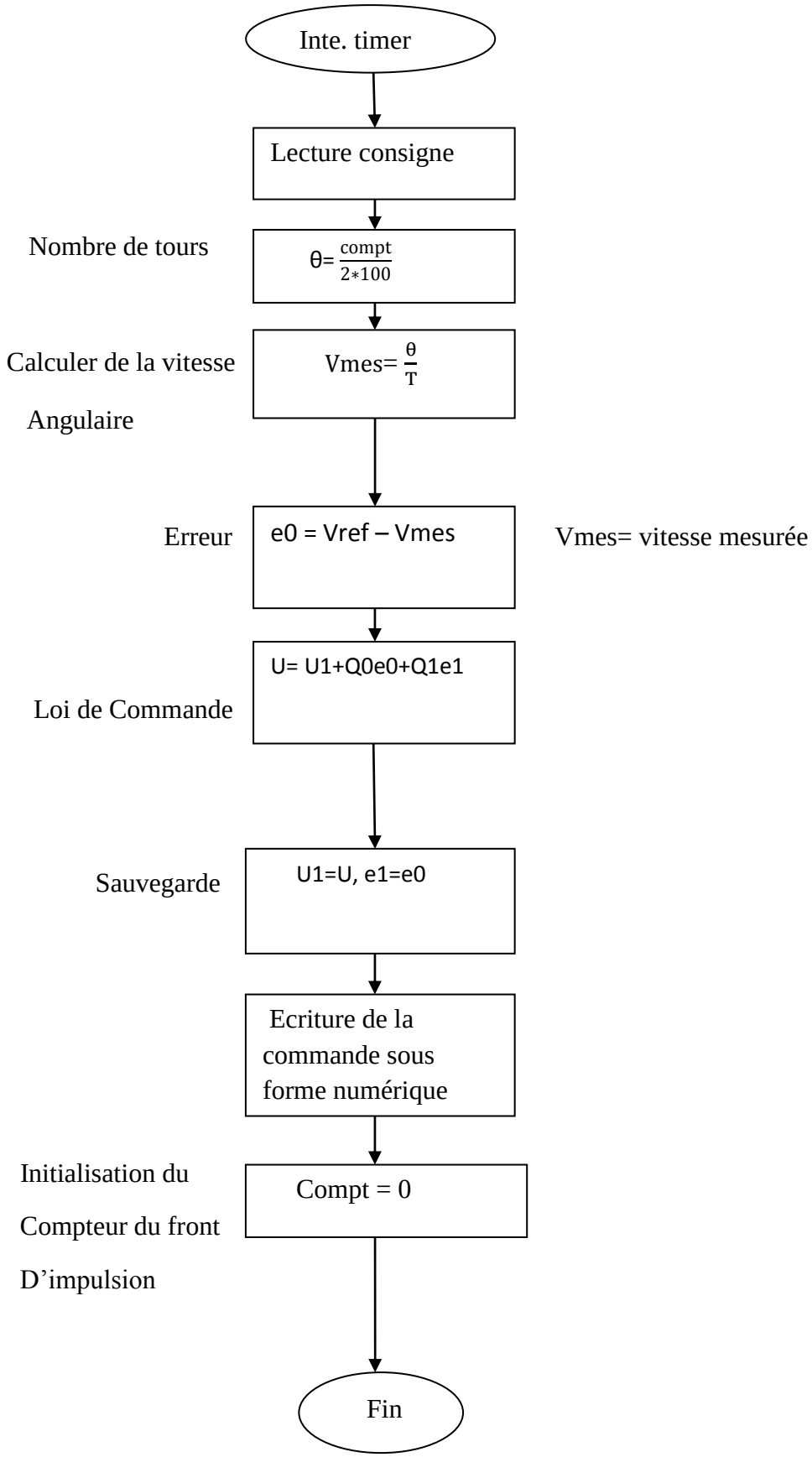


Figure III.9.*Organigramme de Programme de la génération de la commande*

III.5.3.Les étapes de la réalisation

III.5.3.1.Introduction

Après donner les schémas électriques et synoptiques de notre projet

On passe l'étape de la réalisation ou le test pratique.

Bien sûr à l'aide de nos composants électriques qui relient le MCC et le pic

On passe par 07 étapes nécessaires pour déterminer l'algorithme ou le parcours de la commande de la vitesse de MCC.

Ces étapes sont séquentielles en ordre comme on a montré au schéma synoptique

On explique bien nos étapes en suivant :

Etape1 :

Lire la vitesse de référence a travers l'ADC (convertisseur analogique numérique)

Comme on choisir la pâte RA0 comme une entrés de référence dans le PIC16F877A

ETAPE 2 :

Mesurer le nombre des tours qui provient sous forme des impulsions généré par l'encodeur incrémentale via la pâte RB0 de porte B de pic 16f877a qui fonctionne comme interruption d'entrée.

ETAPE 3 :

Calculer la vitesse pendant une durée fixe (10ms) .cette durée est généré par TIMER CALCULATOR

ETAPE 4 :

Appliquer la loi de commande d'un régulateur du régulateur PI

ETAPE 5 :

L'écriture (envoie) de la commande

ETAPE 6 :

Revenir l'étape plusieurs fois jusqu'à que l'erreur tant vers zéro.

ETAPE 7 :

En refait les mêmes étapes si la vitesse de référence change

Donc après citez toutes les étapes nécessaire pour donner une commande d'un MCC sous l'implémentation de correcteur PI .on va discuter chaque étape pour expliquer mieux notre projet.

III.5.4. ETAPE 1

La première étape est de configurer l'ADC dans le pic 16F877A pour définir la tension de référence qui représente la vitesse du moteur

III.5.4. 1.Configuration de PIC 16F877A

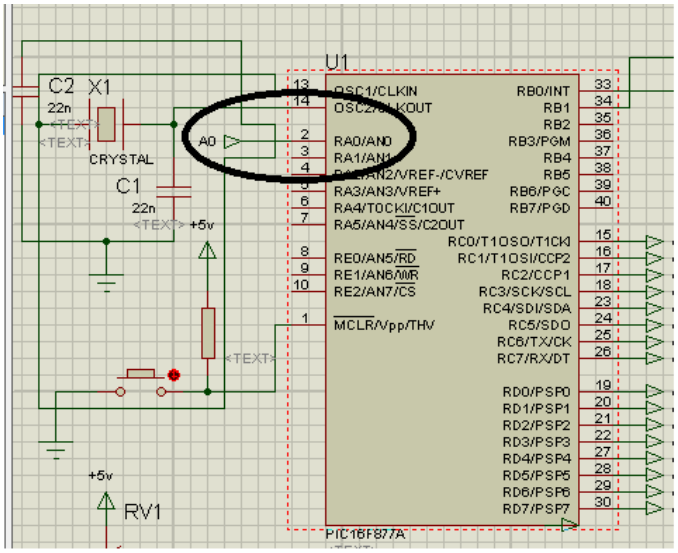


Figure III.10. *L'entrée d'ADC RA0*

III.5.4.2.configuration du convertisseur A/D

Un convertisseur analogique-numérique (CAN, parfois *convertisseur A/N*), ou en anglais ADC (pour Analog to Digital Converter) ou plus simplement A/D, est un montage électronique dont la fonction est de traduire une grandeur analogique en une valeur numérique(codée sur plusieurs bits), proportionnelle au rapport entre la grandeur analogique d'entrée et la valeur maximale du signal.

Nous allons explorer le principe de fonctionnement du convertisseur A / D:

Le convertisseur analogique-numérique convertit la tension analogique en nombres binaires. Ces nombres binaires peuvent être de longueur différente - 2, 4, 8, 10 bits.

Nous avons configuré le convertisseur du microcontrôleur à travers le registre ADCON1 (ADCON1=143) de telle sorte que :

1	0	0	0	1	1	1	1
Bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

- La première ligne du microcontrôleur de port A: RA0 est configure comme entrée analogique réserver à la consigne (M0=5Vmax)
- Les deux lignes de ports A ; RA2, RA3 comme entrées de tension de références (RA2= 0V, RA3= 5V)
- la résolution du convertisseur est 10 bits cela signifie que 5V est représentée par 1111111111 et le pas du convertisseur est donc de 0.048V qui provient du fait que : $\frac{5V}{1023} = 0.048$ c-à-d :

0V ⟶ 0000000000

0.048V ⟶ 0000000001

0.096V ⟶ 0000000010

2.5V ⟶ 01111111111 .

5V ⟶ 1111111111

Après la configuration on passe à une démonstration sur proteus

III.5.4.3.programme du convertisseur A/D sur mikro c:

```
int ref;      // déclaration
void main(){
 ///////////////initialisation des ports/////////////////////////////////
TRISA.F0=1; // A0 ENTREE ADC 5REFERENCE
ADCON1= 142; /*AN0,AN1 entrées analogique
        /*  AN2=RA2 ;AN3=RA3
          AN7--AN4 entrées digitales */
TRISD=0;// PORT D SORTIE
//TRISC=0;// PORT D SORTIE
////////////////////////////////////////////////////
while(1){
```

```
ref = ADC_Read(0);      //lecture de la valeur  référence
PORTD  =ref ;    //ecriture
//PORTC  =ref>>8 ;
}
}
```

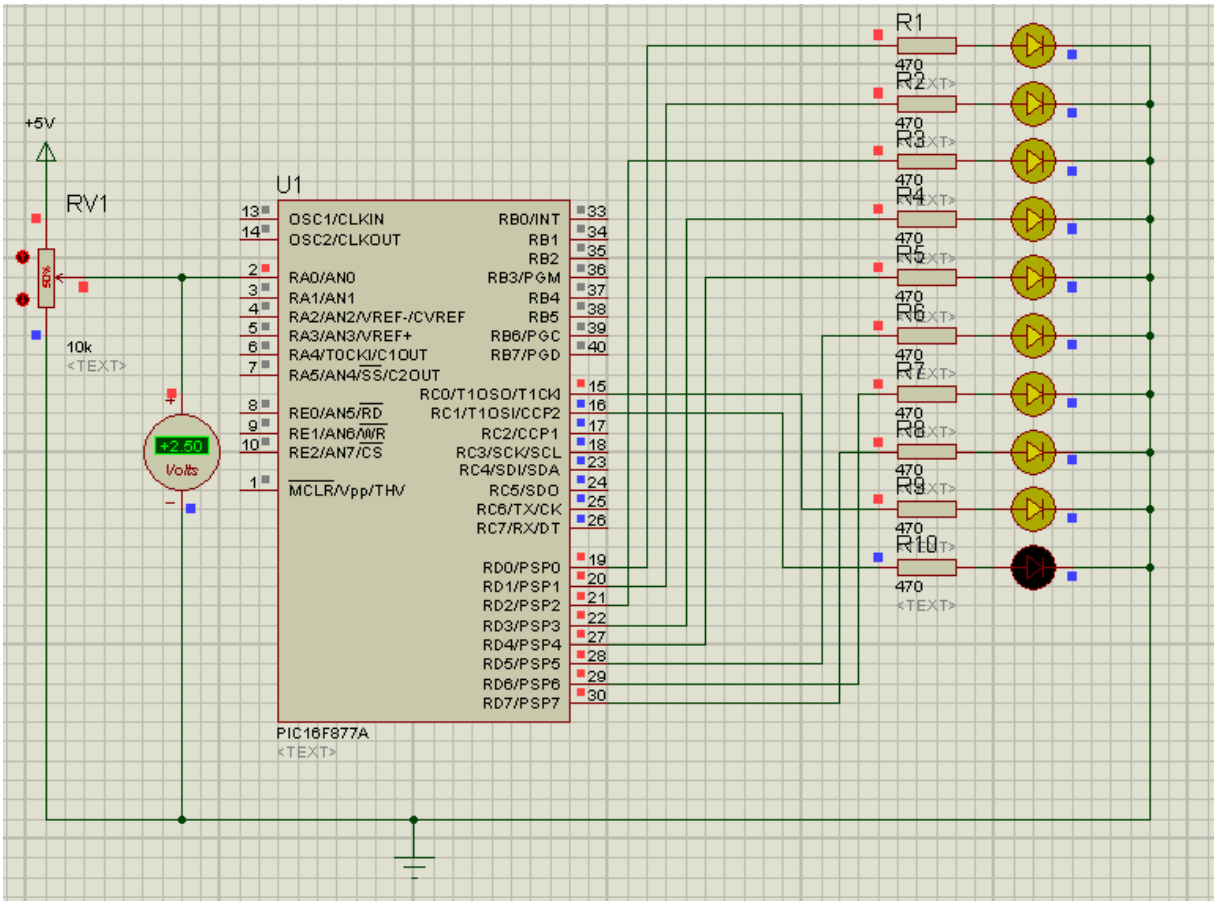


Figure III.11 .*la conversion analogique numérique pour 2.5 v.*

2eme essaie :

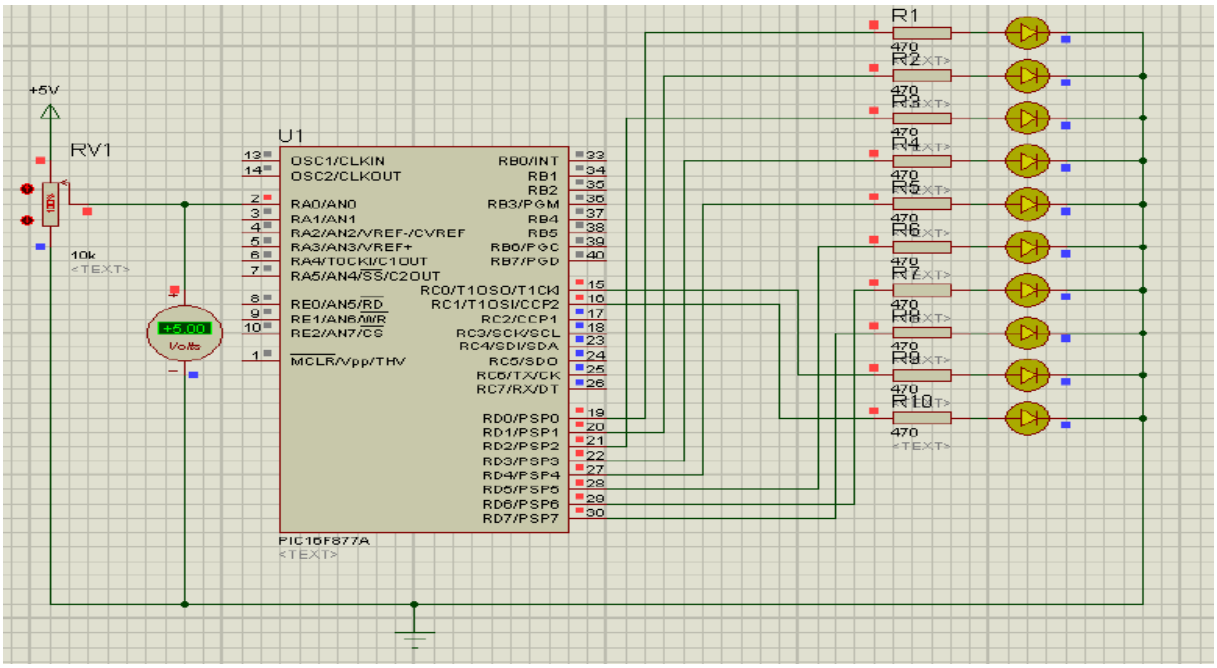


Figure III.12. *La conversion analogique numérique pour 5v.*

III.5.4.4.éssaie pratique du convertisseur A/D

On a fait des essaies pratiques du l'ADC (analog/digital/converter) et pour afficher la sortie numérique ou le résultat de la valeur analogique par les LEDs.

On obtient les resulats suivantes :

Parmi une entrés analogique qui apparierais sur les images suivantes :

I.La première essaie pour 2V donne le résultat sur la plaque : 0110011100

PS : La lecture du résultat se lit de haut vers le bas

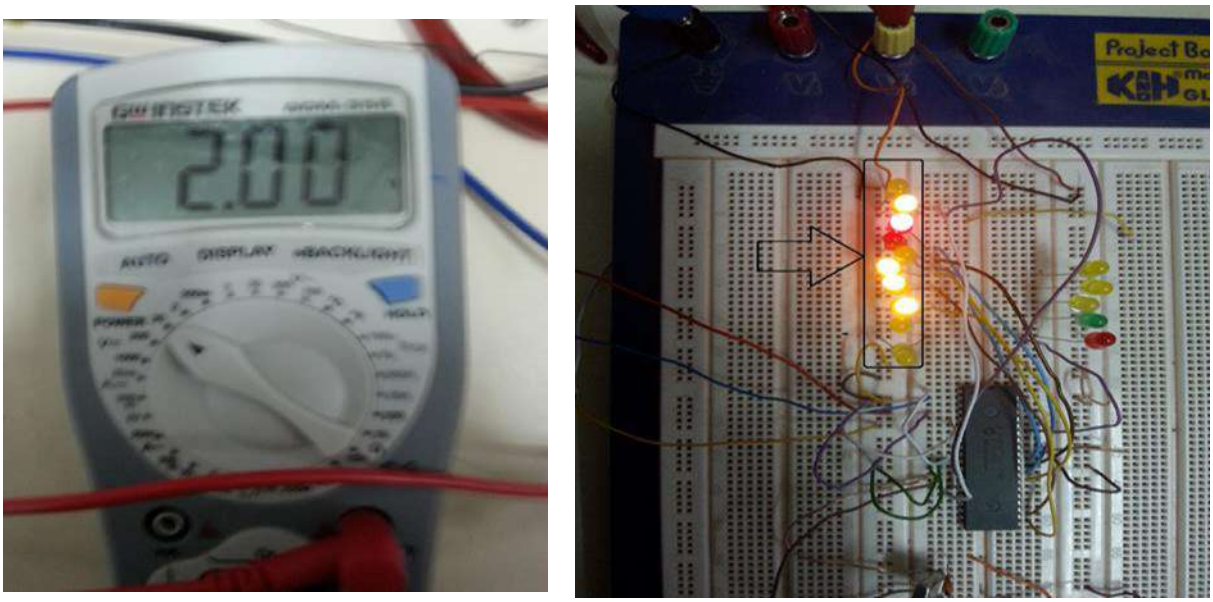


Figure III.13. *Test d'ADC pour 2v.*

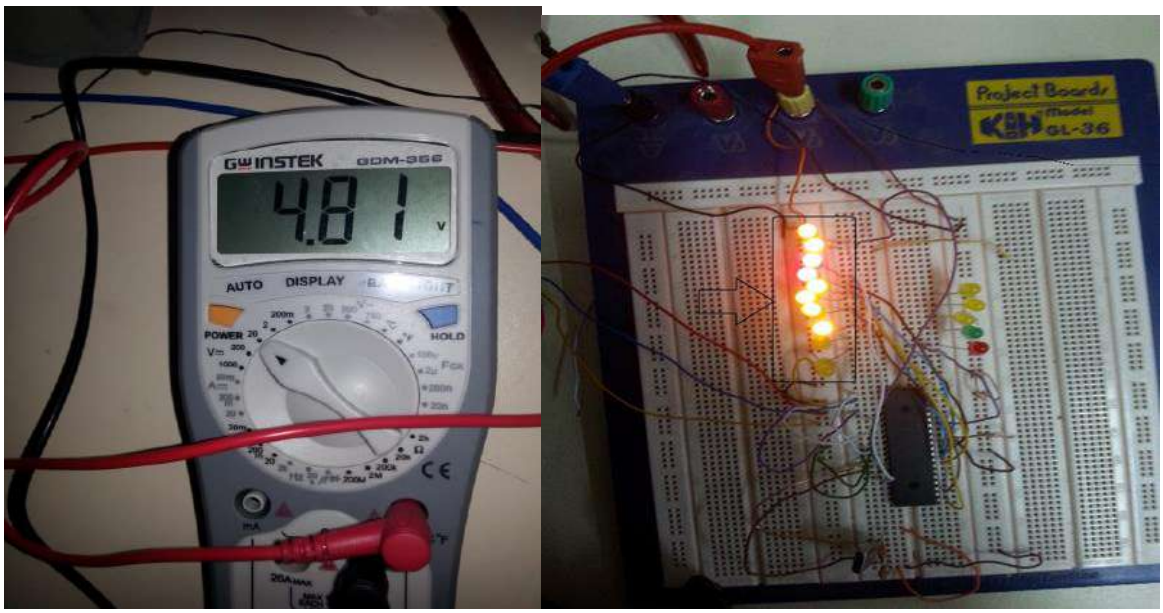


Figure III.14. *Test d'ADC pour 4.81v.*

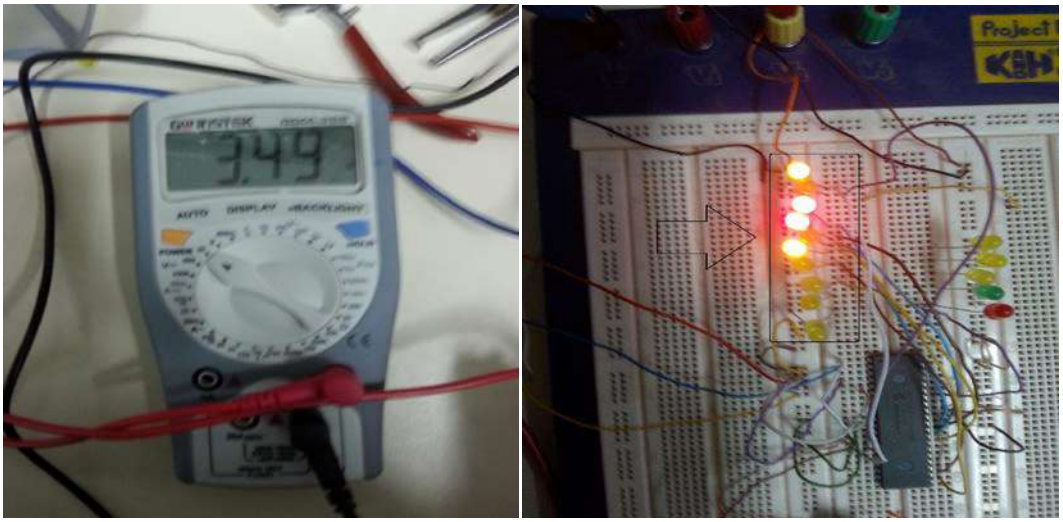


Figure III.15. *Test d'ADC pour 3.49v.*

III.5.5. ETAPE 2

Mesurer le nombre de tour via la pâte RB0

III.5.5.1.Interruption RB0 et RPM

Introduction :

le PIC Microcontrôleur se compose de matériel et de logiciels Interruptions.

Parmi ces interruptions on configure le porte B et partiellement RB0 comme une entrés pour mesure les impulsions générés par l'encodeur.

Les caractéristiques de l'interruption externe que nous utilisons :

Une entrée (selon la façon dont nous avons mis en place) sur le RB0 déclenche l'interruption, soit le signal passe de 0 à 1, appelé le front montant, ou que le signal passe de 1 à 0, appelé le front descendant.

Ceci définit le drapeau d'interruption, et si l'interruption est activée (et nous ne sommes pas déjà dans une interruption), le microcontrôleur va au sous-programme D'interruption.

L'utilisation du RB0 / Int à doit être clair que l'utilisation de la fonction RB0 / Int d'un PICmicro nécessite le réglage de la:
GIE (7): mis à 1 pour activer les interruptions globales
INTE (4): mis à 1 pour permettre des interruptions sur la broche RB0
PEIE (6): pour désactiver d'autres interruptions préférable
et l'ensemble du bit INTEDG (6) du registre de OPTION_REG simplement pour sélectionner le front montant ou descendant du signal.

Lorsque l'interrupteur est enfoncé RB0 va de 0 à 1 provoquant une interruption qui appelle la routine d'interruption (). L'incrément de routine d'interruption de la valeur de la DROP

INTCON register

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

GIE	enable/disable unmasked interrupts
PEIE	enable/disable unmasked periferal interrupts
T0IE	enable/disable Tmr0 interrupts
INTE	enable/disable RB0/Int interrupt
RBIE	enable/disable RB port change interrupt
T0IF	Tmr0 register overflow
INTF	RB0/Int interrupt occurred
RBIF	one of the PORTB (rb4...rb7) has changed state

OPTION_REG register

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	INTEDG						

INTEDG	rising edge or falling edge

. ***III.5.5.2.*Program de RB0 sur mikroc**

```
////////////////////// Initialisation LCD ////////////////////////
sbit LCD_RS at RC4_bit;
sbit LCD_EN at RC5_bit;
sbit LCD_D4 at RC0_bit;
sbit LCD_D5 at RC1_bit;
sbit LCD_D6 at RC2_bit;
sbit LCD_D7 at RC3_bit;
sbit LCD_RS_Direction at TRISC4_bit;
sbit LCD_EN_Direction at TRISC5_bit;
sbit LCD_D4_Direction at TRISC0_bit;
sbit LCD_D5_Direction at TRISC1_bit;
```

```
sbit LCD_D6_Direction at TRISC2_bit;
sbit LCD_D7_Direction at TRISC3_bit;
////////////////////////////////////////////////////////
void interrupt() {
     PORTB.RB1=~PORTB.RB1;
     counter++;          // comptage  de la valeur de révolution
    INTCON.INTF = 0;
    PORTB.B1=0 ;
  }
void main ()
{
 INTCON.GIE=1; // ACTIVATION DES INTERRUPUTION GLOBALE
 INTCON.PEIE=1; //ACTIVATION DES INTERRUPTION DES PERIPHERIQUES
INTCON.INTE=1; //  ACTIVATION DE L'NTERRUPTION  int_rb0
INTCON.INTF = 0;  // MISE A ZERO DU DRAPEAU
 ///////////////initialisation des ports/////////////////////////////////
TRISB=0b00000001;// PORT B SORTIE    SAUF PIN B0 EST ENTREE
TRISC=0;// PORT C SORTIE
///////////////AFFICHAGE ////////////////////////
while(1){
PORTB.RB1=1 ;
Lcd_Init();                        // Initialize LCD
Lcd_Cmd(_LCD_CLEAR);               // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);        // Cursor off
  IntToStr(counter, txt);
  Lcd_Out(1,1,"Compteur=");
  Lcd_Out_Cp(txt);                 // Write text in first row
```

```
delay_ms(1);
  PORTB.RB1=0 ;
}
}
```

Après la création de ce programme.

On créer un schéma par le proteus.

Le schéma suivant présente l'interruption RB0 qui est généré par le DC clock

Donc on compter 62 comptes par 1.25v.

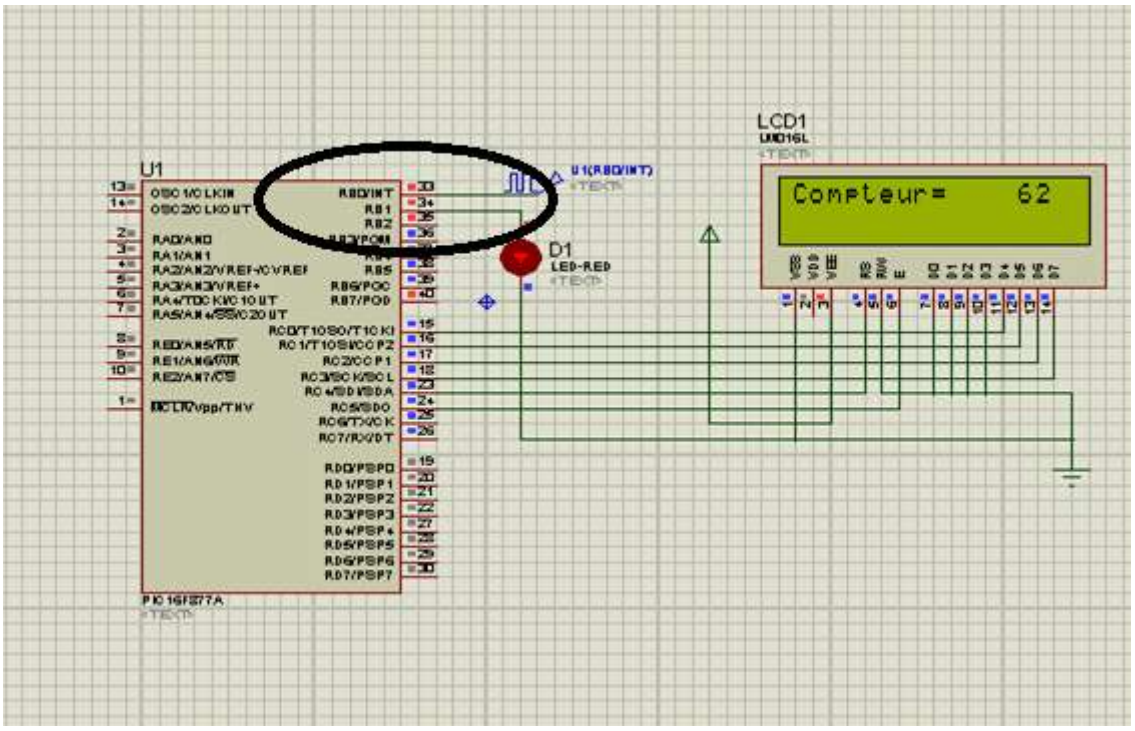


Figure III.16.*test par simulation pour Vref=1.25v.*

III.5.5.3. Essaie pratique Interruption RB0

Après la simulation sur logiciel protes (parité software).on essayeras de faire la même expriment (partie hardware). L'image suivante illustre un LED qui clignote a l'aide d'un signal qui arrive par le port RB0.

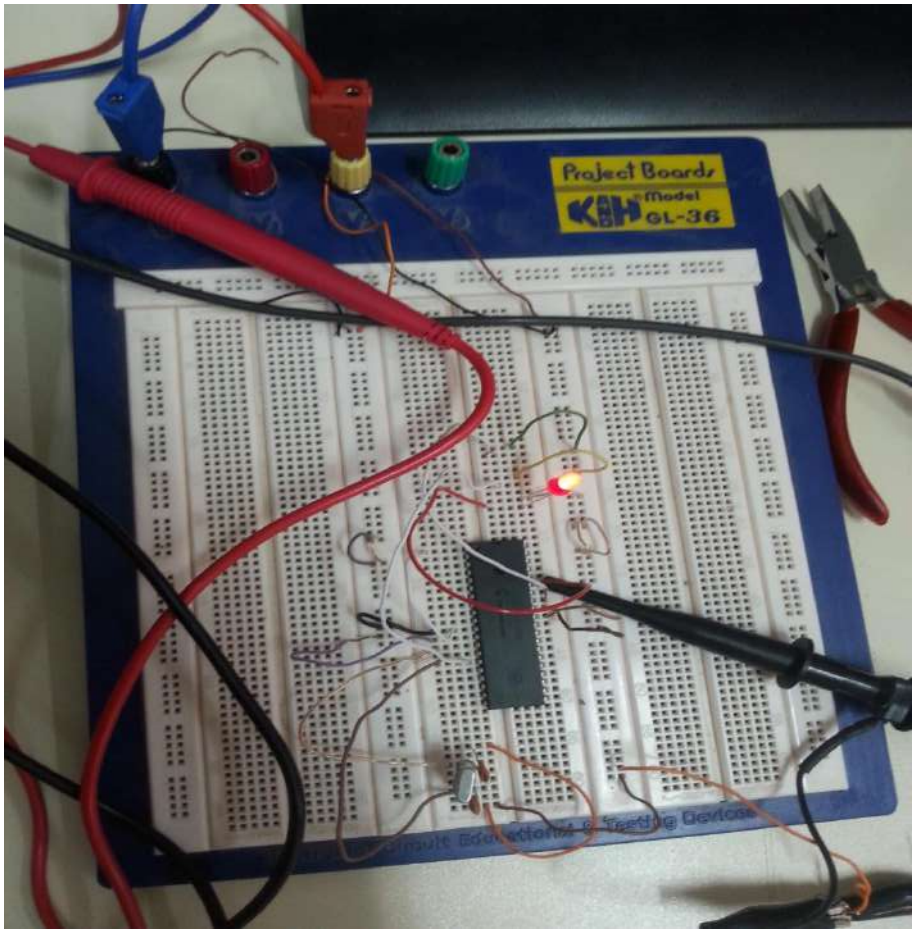


Figure III.17.*test d'une interruption RB0.*

III.5.6.ETAPE 3

Calculer la vitesse pendant une durée fixe (10ms)

III.5.6.1. Timer1 interrompre

Le module Timer1 est un 16-bit temporisateur / compteur composé de deux registres de 8 bits (TMR1H et TMR1L) qui sont en lecture et écriture. La paire de registres TMR1 (TMR1H: TMR1L) incrémente de 0000h à FFFFh et roule à 0000h.

L'interruption TMR1, si elle est activée, est générée en cas de débordement qui est verrouillé dans le bit de drapeau d'interruption, TMR1IF (PIR1 <0>). Cette interruption peut être activée / désactivée en réglant / compensation TMR1 interruption bit de validation, TMR1IE (PIE1 <0>).

L'intervalle de temps est mesuré par le timer de microcontrôleur ou ce dernier déclenche une interruption interne à chaque T. (T : période d'échantillonnage). Dans notre cas on a travaillé avec le timer1du 16 bits parce qu'il est plus précis par rapport au timer0 qu'il a 8 bits.

L'équation qui permet de calcule T est donner comme suite :

$T = 4 * T_h *$ pré division $*(65535-VI)$ **(III.3)**

Ou : T_h : période d'horloge

VI : valeur initiale

On a une fréquence de 8MHz ⟶ donc $T_h = 1/8*10^6$

Après l'introduction sur le timer1 .on passe a créer un programme sur MIKROC

III.5.6.2.programme de TMR 1 en MIKROC

```
char txt[8];
float Tech=10ms;//    période d'échantillonnage
////////////////// initialisation LCD ///////////////////
sbit LCD_RS at RC4_bit;
sbit LCD_EN at RC5_bit;
sbit LCD_D4 at RC0_bit;
sbit LCD_D5 at RC1_bit;
sbit LCD_D6 at RC2_bit;
sbit LCD_D7 at RC3_bit;
//
sbit LCD_RS_Direction at TRISC4_bit;
sbit LCD_EN_Direction at TRISC5_bit;
sbit LCD_D4_Direction at TRISC0_bit;
sbit LCD_D5_Direction at TRISC1_bit;
sbit LCD_D6_Direction at TRISC2_bit;
sbit LCD_D7_Direction at TRISC3_bit;
void interrupt() {
 if (PIR1.TMR1IF)        //
   {
```

```
    T1CON.TMR1ON=1;
    // Fréquence d'horloge 8Mhz
    //VI= 3CAF pour 100ms
    //VI=EC77 pour 10ms
    // Tech=100ms
    TMR1L=0x77 ;       //V.I poids faible
    TMR1H=0xEC ;       //V.I poids fort
    // Tech=10ms
    PORTB.RB1=PIR1.TMR1IF;
    // delay_ms(10);    // POUR 100ms
    delay_ms(1);    // POUR 10ms
    PIR1.TMR1IF = 0;
    PORTB.RB1=PIR1.TMR1IF;
    }
  }
void main () {
 INTCON.GIE=1; // ACTIVATION DES INTERRUPUTION GLOBALE
 INTCON.PEIE=1; //ACTIVATION DES INTERRUPTION DES PERIPHERIQUES
 ////////////////////////////INTERRIPTION RB0/////////////////////////////
INTCON.INTE=1; //  ACTIVATION DE L'NTERRUPTION  int_rb0
INTCON.INTF = 0;   // mise a zero le drapeau de l'interruption de rb0
//////////////////timer0////////////////////////////////////////////////
PIE1.TMR1IE=1;//ACTIVATION L'INTERRUPTION DU TIMER1
PIR1.TMR1IF=0; // LA %ISE A 0 DU DRAPEAU DU TIMER1
T1CON=0b00100001;  // CONFIGURATION DU TIMER 1
//T1CON.TMR1ON=1;    activation du  timer1
//T1CON.TMR1CS = 0;SELECTION DE L'HORLOGE INTERNE DU UC
```

```
//T1CON.T1OSCEN=0  LA MISE EN ARRET DE L'OSCLLIATEUR EXTERNE
//T1CON.T1CKPS1=1;   1:4 prescaler
//T1CON.T1CKPS0 = 0; 1:4 prescaler
// T1CON.TMR1GE=0;
//T1CON.T1GINV=0;
TMR1L=0x77 ;       //V.I poids faible
TMR1H=0xEC ;       //V.I poids fort                // F_int=8Mhz
//////////////////////////configuration convertisseur////// ///////////
TRISA.F0=1; // A0 ENTREE ADC 5REFERENCE
ADCON1= 143; // configuration AN0 entrée analogique
 ///////////////initialisation des ports////////////////////////////////////
TRISA.F0=1; // A0 ENTREE ADC 5REFERENCE
ADCON1= 142; // configuration AN0 entrée analogique;AN2=réf-   ;AN3=réf+
TRISB=0x00; //PORT b SORTIE
TRISD=0;// PORT D SORTIE
while(1){
  Lcd_Init();                    // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);              // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);         // Cursor off
  IntToStr(Tech, txt);
  Lcd_Out(1,1,"Tech=");
  Lcd_Out_Cp(txt);               // Write text in first row
  Lcd_Out_Cp("ms");
 delay_ms(100);
}
}
```

Après la création de notre fichier TMR1.HEX on passe à créer un schéma correspondant a ce programme.

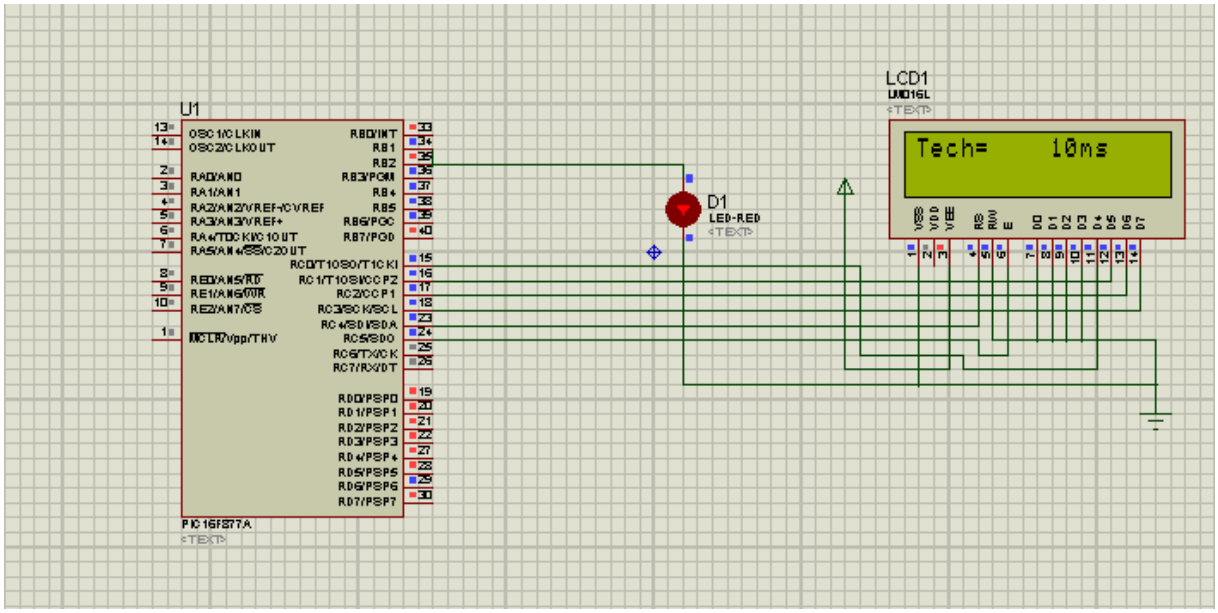


Figure III.18. *Schéma électrique du timer1.*

III.5.6.3.essai pratique sur L'interruption TMR1

Après la simulation de TMR1.on essayons de donner la réalisation qui s'apparaît :

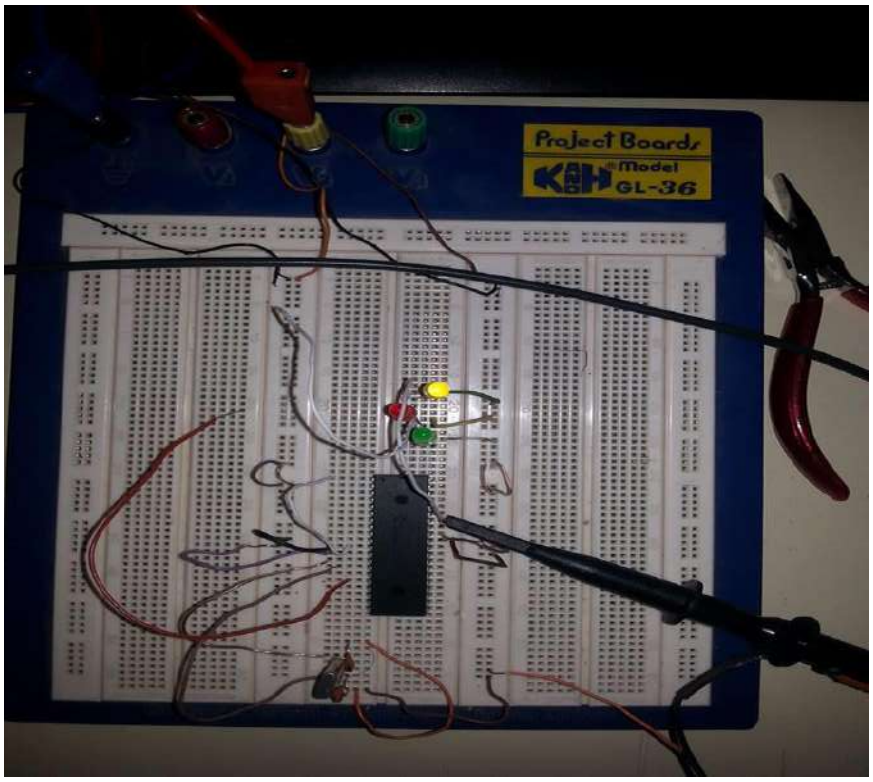


Figure III.19. *Test de TMR1***.**

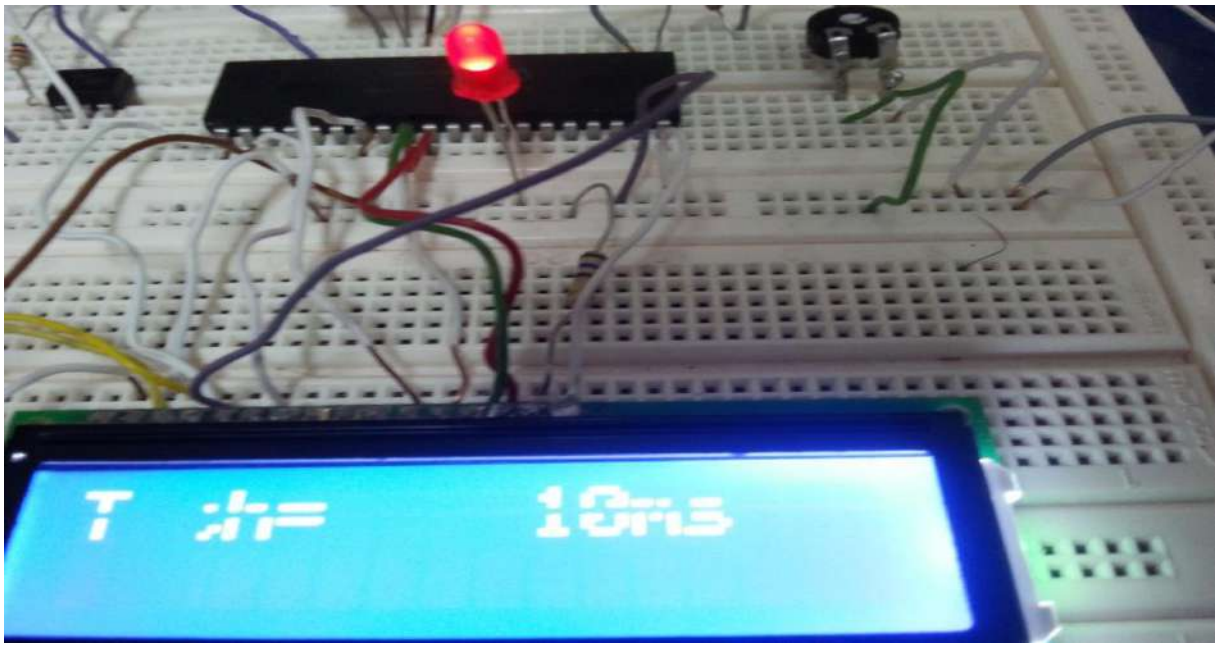


Figure III.20.*l'affichage de TMR1 sur LCD.*

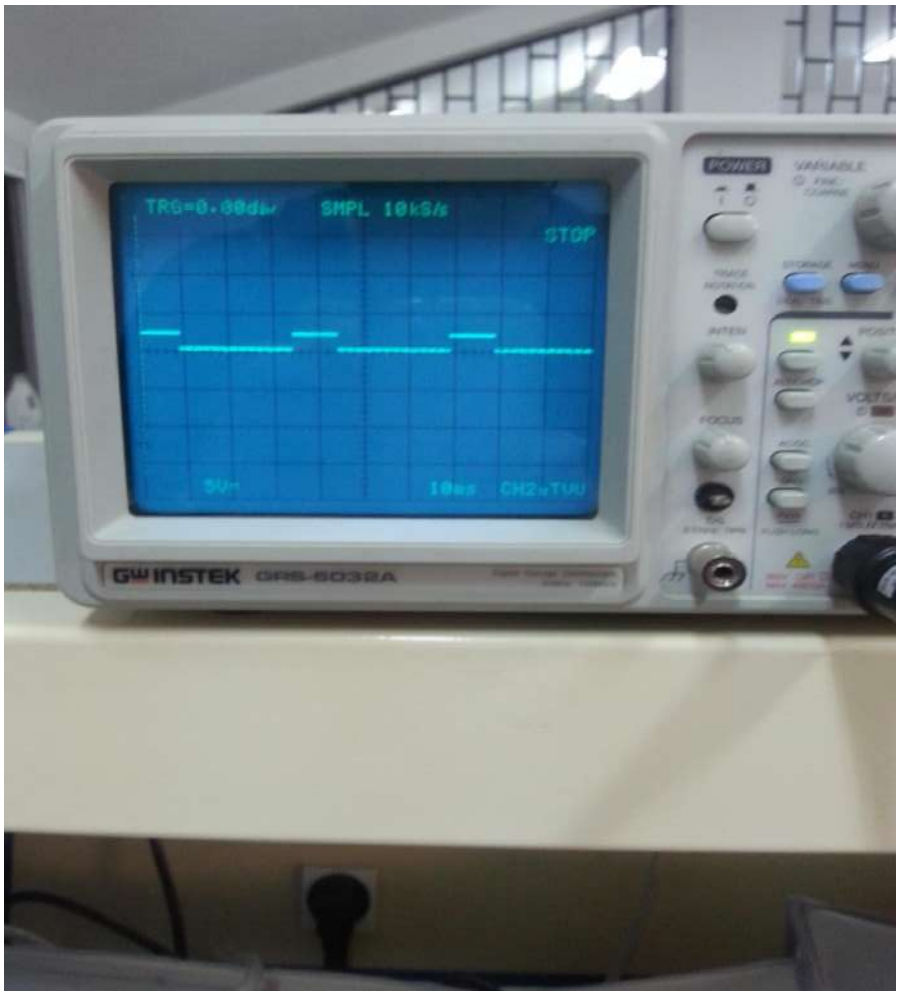


Figure III.21.*l'affichage TMR1 sur l'oscilloscope.*

III.5 .7.l'encodeur

Notre modèle est comporte 200 révolutions.

Le rôle de l'encodeur est de mesurer la vitesse du moteur

On essaye de faire une simulation de l'encodeur pour illustré leur caractéristiques

III.5.7.1.SIMULATION ET ESSAIE PRATIQUE

On a besoin d'un programme sur MIKROC pour l'encodeur .car on peut le visualiser a travers n'importe quelle modèle de MCC.

On essaye de réalisé un encodeur de 200 révolutions et on obtient les résultats de simulation comme celle-ci :

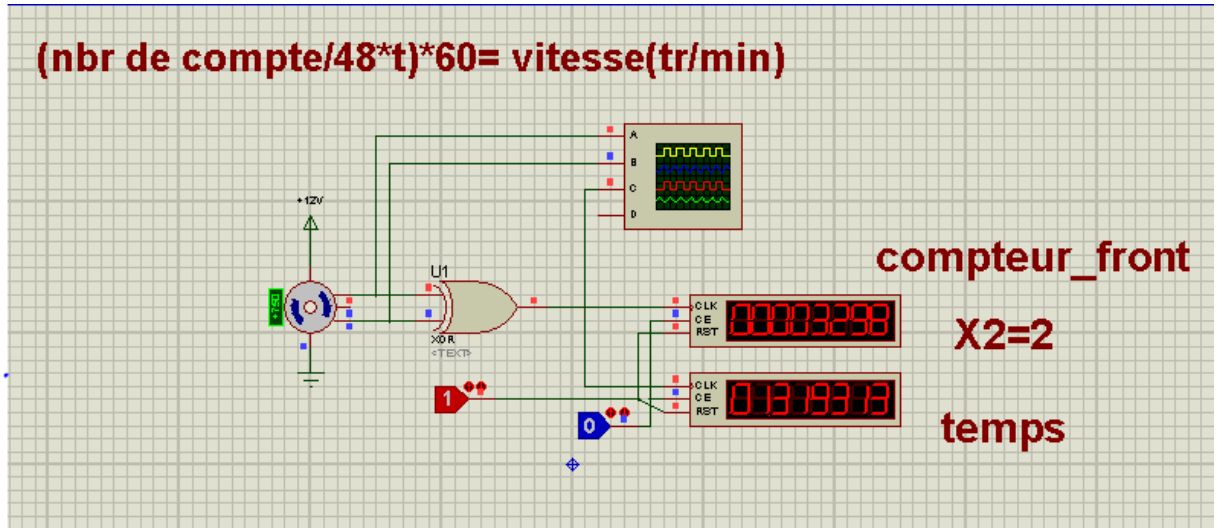


Figure III.22.*Schéma de simulation d'un encodeur 200 révolutions.*

D'après le schéma on insérant 02 compteurs :

- Le premier pour compter les impulsions généré par l'encodeur.
- Le deuxième pour compter le temps de comptage nécessaire pour le premier compteur.

Donc on a le résultat suivant :

Peut mesurer la vitesse de moteur par :

RPM = (nombre de compte /48*t)*60

Donc

(3298/48*1.3)*60=3170tr/min

III. 5.7.2.essaie pratique

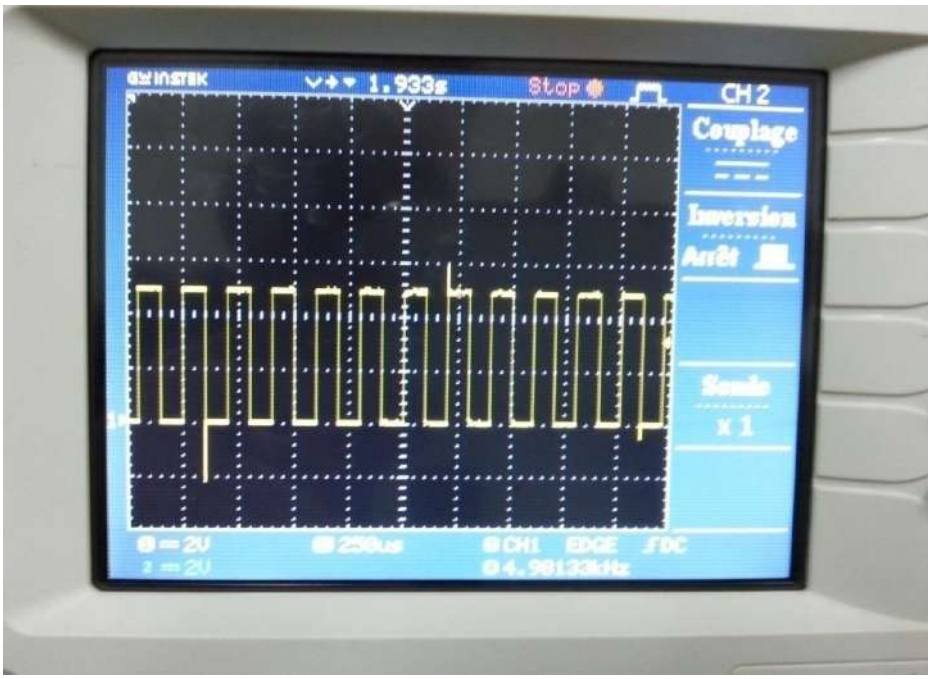


Figure III.23.*l'affichage des comptes sur l'oscilloscope***.**

Dans cette figure de l'oscilloscope .on constatant le graphe qui alterne à chaque 0.2 ms

Donc après le formula 0.2*100=10ms.

Et 3000rpm=50rps=1tourne chaque 20ms

On déduire qu'à chaque cycle représente une tourne

III.5.7.3.calcule de la vitesse

La vitesse nominale du notre moteur est de 3000 tr /min, nous avons donc donné la tension 12V pour présenter cette vitesse :

$$Vn = \frac{E.3000}{1023} \text{ tr /min} \qquad \textbf{(III.4)}$$

Ou :

E : consigne après conversion (consigne numérique)

Vn : vitesse nominale numérique

Exemple :

Si la consigne 6V alors

$E=(0001111110)_2 \longrightarrow (E=510)_{dec}$

D'où $V=\frac{510*3000}{1023}$ = 1495 tr/min

Dans ce schéma on voie que si Unum= $(1111111111)_2$, Alors Uana= 12V.

III .5.7.5. Mesure de la vitesse du moteur

Le moteur qu'on a utilisé est doté d'un encodeur incrémentale de deux rangées, chaque rangé compte 100 révolution, les deux sorties de ce encodeur sont introduit dans une porte XOR afin d'avoir un seul train d'impulsion (x2=2).

D'après la création de la vitesse sous logiciel MIKROC et l'attribution sous logiciel PROTEUS

La sortie de celle-ci est reliée à l'entrée RB0-int du microcontrôleur (utilisé comme entrée d'interruption externe) dans le but de comptée le nombre de fronts. De ce train d'impulsion le compte va nous permettre de savoir le nombre de tour fait par le moteur par l'application de la formule suivante :

$$\Theta° = \frac{\text{nombre de fronts}}{2*100} 360° \qquad \textbf{(III.5)}$$

Par exemple le moteur fait 1 tour, alors le nombre de fronts sera 200 ce qui donne ɵ°=360°

- Pour calculer maintenant la vitesse angulaire on prend le nombre de tour trouvé précédemment et on le divise sur un intervalle de temps fixe. (T=10ms dans notre cas) d'après l'équation suivante :

$$\frac{d\theta}{dt} = \frac{\theta}{T} \qquad \textbf{(III.6)}$$

III.5.7.5.1. programme pour la vitesse sur MIKROC

```
# include "lcd_setup.c"

# include "Init_RB0_TIMER.c"

 int counter,tcounter;

 float tour;
```

```
 float T;
 float TPM;
 char txt1[10],txt[8];
/*void Init_RB0_TIMER(void)
{                                                    .
INTCON.GIE=1;        // ACTIVATION DES INTERRUPUTION GLOBALE
INTCON.PEIE=1;       //ACTIVATION DES INTERRUPTION DES PERIPHERIQUES
INTCON.INTE=1;       //  ACTIVATION DE L'NTERRUPTION  int_rb0
PIE1.TMR1IE=1;      // ACTIVATION L'INTERRUPTION DU TIMER1
INTCON.INTF = 0;   // mise a zero le drapeau de l'interruption de rb0
PIR1.TMR1IF=0;     // mise a zero le drapeau de  du timer1
 T1CON        = 0x01;    // T=10ms
 TMR1H        = 0x59;
 TMR1L        = 0xFC;
//Prescaler 1:4; TMR1 Preload = 3036; Actual Interrupt Time : 20 ms
 T=4.*(1/20000000.)*(4.)*(65535.- 23036.);
}   */
// programme d'interruption//////
void interrupt() {
if (INTCON.INTF)
  {
   counter++;//bascule de l'état
   INTCON.INTF = 0;   // mise a zero le drapeau de l'interruption de rb0
```

```
   }
else if (PIR1.TMR1IF)
  {
  TMR1H        = 0x59;
  TMR1L        = 0xFC;

  PORTB.B1=PIR1.TMR1IF;
  tcounter=counter;
  tour=tcounter/100.;    // nombre de tour (x2=2, revolution =24)
  TPM= (tour/T)*60.;    // nombre de tour par minute
  counter=0;           // réinitialisation du counter
  PIR1.TMR1IF=0;
  PORTB.B1=PIR1.TMR1IF;
  }
 }
void main(){
Lcd_Init();                    // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);              // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);         // Cursor off
  Init_RB0_TIMER();
  TRISB=0b00000001;// PORT B SORTIE
while(1){
  intToStr(tcounter, txt);
```

```
 Lcd_Out(1,1,"counter=");
 Lcd_Out_Cp(txt);
 floatToStr(TPM, txt1);          // Write text in first row
 Lcd_Out(2,1,"TRM=");
 Lcd_Out_Cp(txt1);
}
}
```

Sa donne les résultats suivantes qui s'apparais :

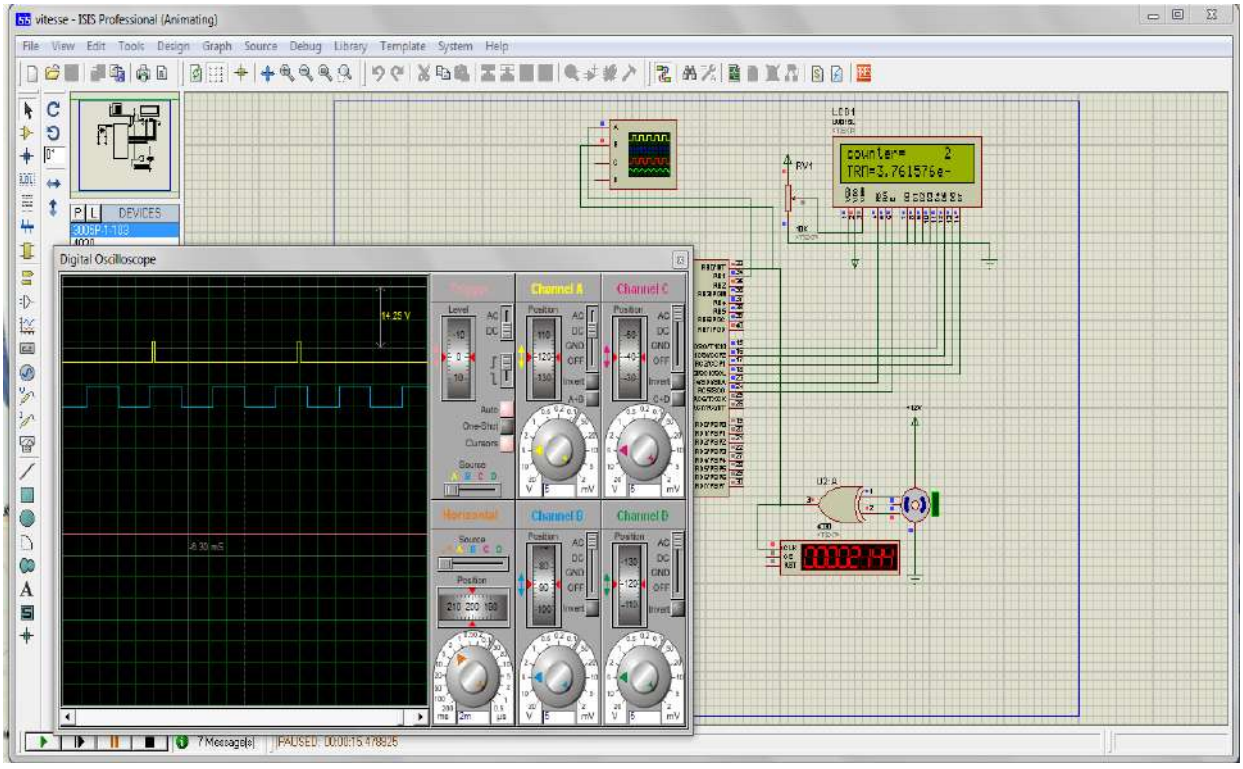
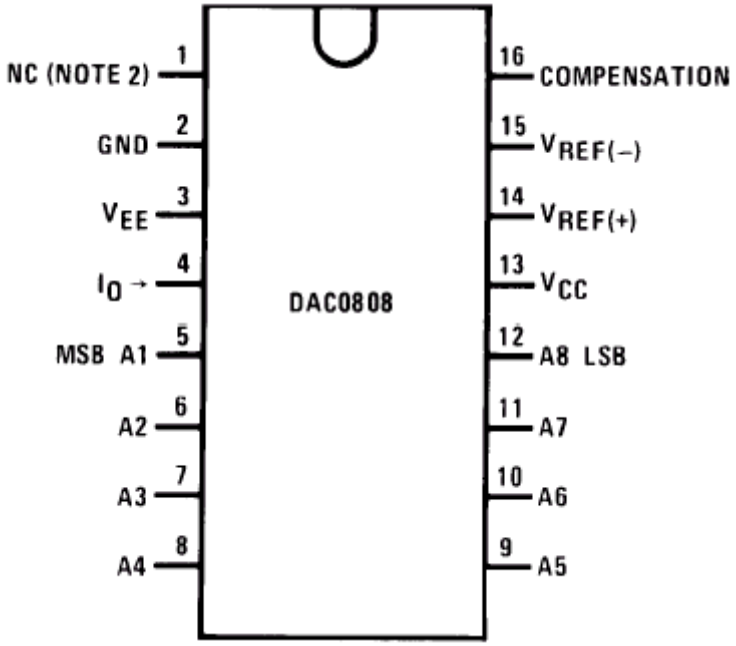


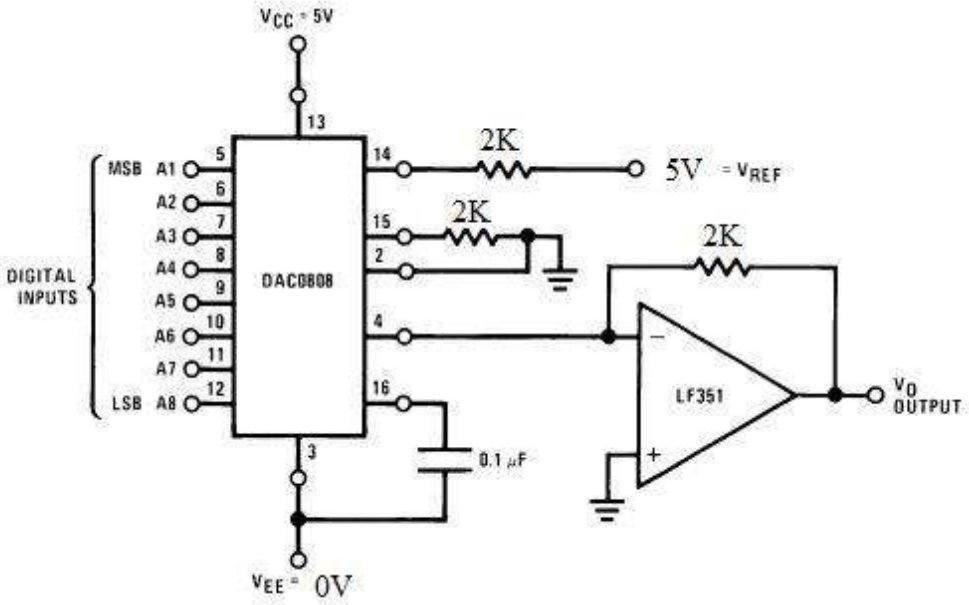
Figure III.24.*Schéma de simulation de calcul de la vitesse.*

III.5.7.5.2. convertisseur numérique analogique

Le DAC0808 est un convertisseur numérique-intégration analogique 8 bits avec sortie courant. La configuration des broches est illustrée dans la figure suivante



Les broches d'entrée numériques sont A1 ... A8, A1 étant le plus important et A8 le bit le moins significatif (les noms choisis ne suivent pas la convention habituelle). Le courant de sortie est prévu sur la broche 4 (I) et doit être converti en tension par un convertisseur courant-tension, par exemple par le schéma suivant:



Notez la résistance 5 kQ appliquée sur la broche 14. La valeur de cette résistance est égale à celle utilisée dans les évaluations de fonctionnement. En effet, le courant de sortie produit par le convertisseur est donnée par:

$$I_0 = K(\frac{A_1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}$$

$$where\ k = \frac{vref}{R14}$$

Où R14 est précisément la résistance appliquée à la borne 14. En sélectionnant une telle résistance égale à celle du convertisseur courant-tension, la tension de sortie Vout est donnée par:

$$Vout = Vref(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A7}{128} + \frac{A8}{256})$$

après le calcul de la vitesse on attribuant un DAC pour convertir les sorties numériques au l'entrés de DAC vers un signal analogique pour conduire le moteur..

Et on ajoutant un lampliope pour convertir le courant vers une tension parce que notre DAC 0808 donne une sortie de courant comme on besoin un entrée de tension sur le moteur

III.5.7.5.3.Programm du convertisseur numérique analogique MIKROC

```
void main()
{
int i=00;   //initialisation
TRISD=0;// PORT D SORTIE

while(1){    //boucle infinie
PORTD  =i;    //écriture
delay_ms(100);
i=i+1;       //incrimination
 if (i==255)   // si PORTD =0b11111111
 i=0;         // recommencement
```

```
    }
}
```

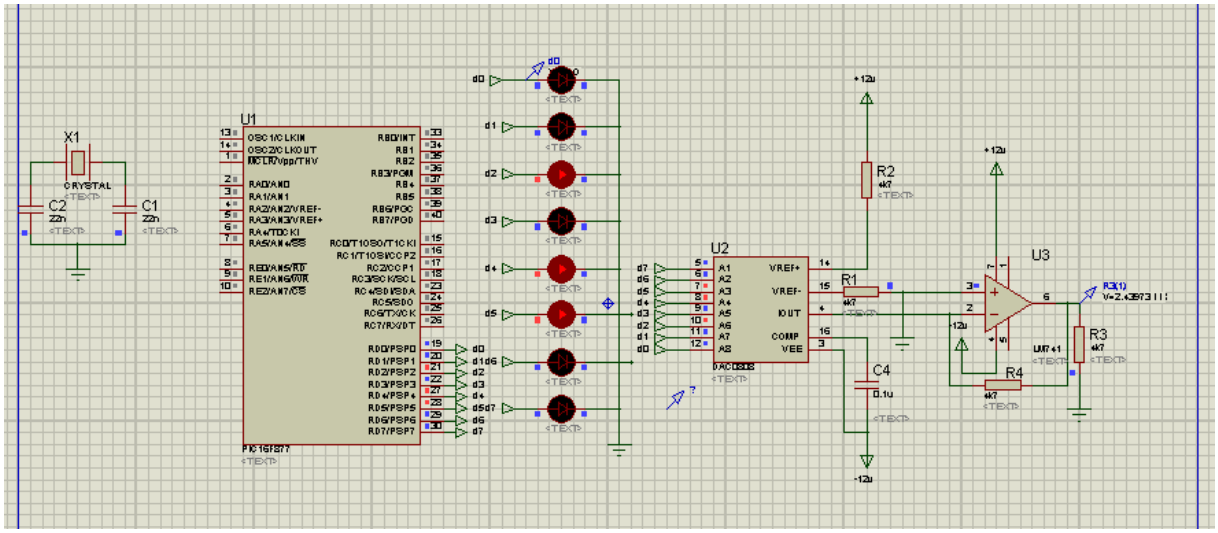


Figure III.25.*Etage de puissance.*

III.5.7.5.4. Essaie pratique du convertisseur numérique analogique DAC

On réalisant le DAC et on fait des essaies pour déduire son fonctionnement et son caractéristique et son résolution

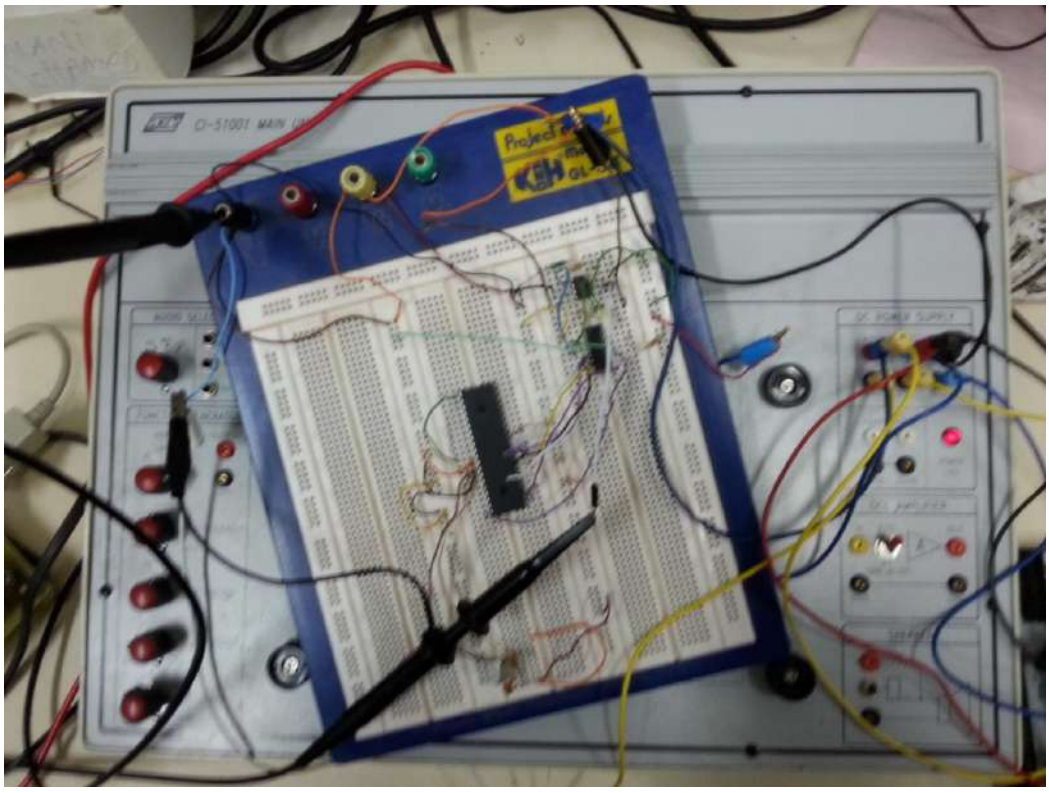


Figure III.26.*Schéma de DAC relié avec ampli ope LF35.1*

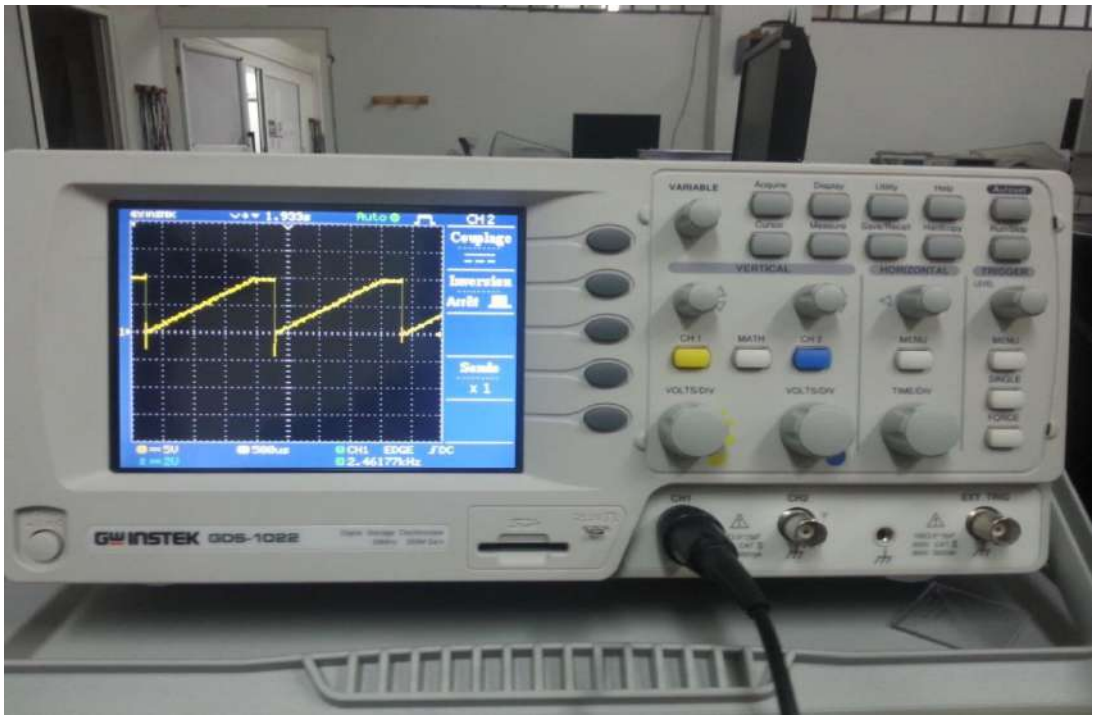


Figure III.27.*resultats de la réalisation sur l'oscilloscope.*

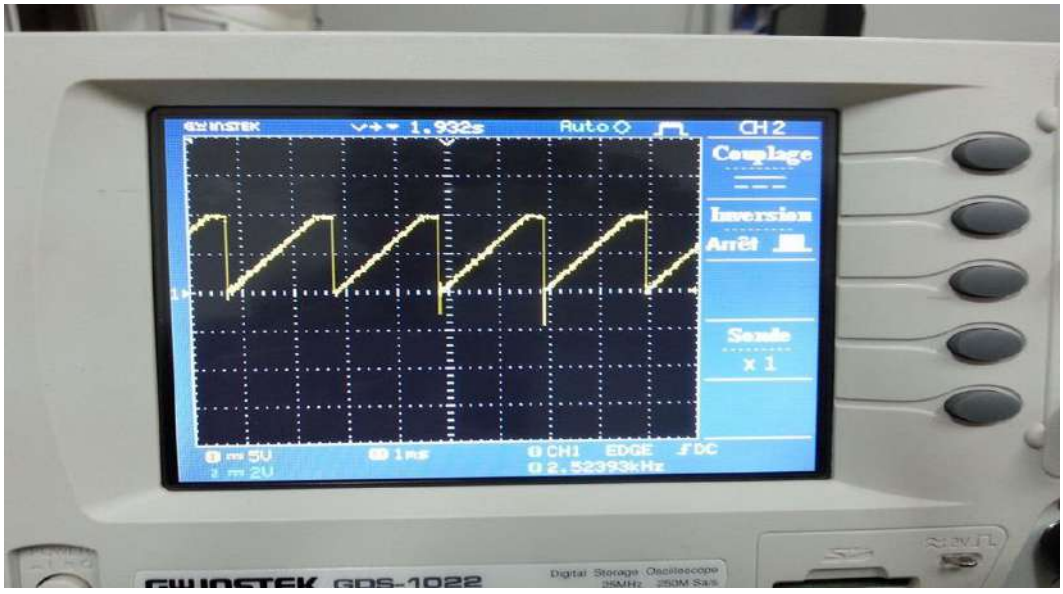


Figure III.28. *Résultats de la réalisation sur l'oscilloscope.*

IV.5.8. le résultat final

Après LA CONFIGURATIONS des moyens nécessaire pour définir la vitesse et sa variation par l'intervention de l'interruption RB0 et TMR1 et ADC et DAC et LCD

Et le rôle de l'encodeur pour offrir la vitesse.

On implémente un régulateur ou correcteur PI pour corriger l'erreur entre les valeurs consignées et les valeurs de référence.

On donne le résultat finale après récolte toutes les moyens nécessaires qui s'apparait ci-dessus :

La première étape est de créer le programme final.

IV.5.8.1.Programmers final avec pid

```
// LCD module connections
sbit LCD_RS at RC4_bit;
sbit LCD_EN at RC5_bit;
sbit LCD_D4 at RC0_bit;
sbit LCD_D5 at RC1_bit;
sbit LCD_D6 at RC2_bit;
sbit LCD_D7 at RC3_bit;
sbit LCD_RS_Direction at TRISC4_bit;
sbit LCD_EN_Direction at TRISC5_bit;
sbit LCD_D4_Direction at TRISC0_bit;
sbit LCD_D5_Direction at TRISC1_bit;
sbit LCD_D6_Direction at TRISC2_bit;
sbit LCD_D7_Direction at TRISC3_bit;
// End LCD module connections
///////////////déclaration des variables/////////////////
char txt[8] ;int i=120;
  long int  counter,tcounter   ;
  float tour;
  float T ,U;
  float TPM;
  float ref  ;
  float Q0,Q1,Q2,m1,m2 ,e2,e1=0.0 ,e0 ,m0;
```

```
 //float Kp=3.,Ki=0.0;
 double float Kp=4.77 ,Ki=390.28 .;
 //double float Kp=0.00952 ,Ki=4.9132 ,.;

// programme d'interruption//////
void interrupt() {
if (INTCON.INTF)
  {
   counter++;//bascule de l'état
   INTCON.INTF = 0;   // mise a zero le drapeau de l'interruption de rb0
  }
else if (PIR1.TMR1IF)
 {
 PORTB.B1=PIR1.TMR1IF;
 TMR1H        = 0x63;
 TMR1L        = 0xC0;

 /////////////// Lecture de la référence ////////////////
   ref = ADC_Read(0);      //lecture de la valeur  référence
   U=ref*1500./1023.;      // converession tension--vitesse

//////////////////calcule de la vitesse//////////////////
 tcounter=counter;
 tour=tcounter/200.;    // nombre de tour (x2=2, revolution =100)
 TPM= (tour/T)*60.;    // nombre de tour par minute
 counter=0;           // réinitialisation du counter
 ////////////////// régulation   ////////////////////////
```

```
 Q0 = Kp + Ki*T/2  ;
 Q1 = -Kp + Ki*T/2  ;
;
   e0= U-TPM;
   m0 = m1+ Q0*e0+Q1*e1;
   m1 = m0;
   //e2=e1;
   e1 = e0;
   PORTD=e0 ;     // commande
 ////////////////// fin de régulation ////////////////////
 PIR1.TMR1IF=0;
 PORTB.B1=PIR1.TMR1IF;
}
 }
void main(){
 INTCON.GIE= 1;       // ACTIVATION DES INTERRUPUTION GLOBALE
 INTCON.PEIE=1;       //ACTIVATION DES INTERRUPTION DES PERIPHERIQUES
 INTCON.INTE=1;       //  ACTIVATION DE L'NTERRUPTION  int_rb0
 PIE1.TMR1IE=1;      // ACTIVATION L'INTERRUPTION DU TIMER1
 INTCON.INTF = 0;   // mise a zero le drapeau de l'interruption de rb0
 PIR1.TMR1IF=0;     // mise a zero le drapeau de  du timer1
 ADCON1= 142; /*AN0,AN1 entrées analogique   /*  AN2=Vss ;AN3=Vdd  */
 T1CON        = 0x11;    // T=2 ms
 TMR1H        = 0x63;
 TMR1L        = 0xC0;
 //Prescaler 1:1; TMR1 Preload = 64736; Actual Interrupt Time : 2 ms
  T=4.*(1/8000000.)*(2.)*(65535.- 25536.);
```

```
  TRISB=0b00000001;// PORT B SORTIE
  TRISA.F0=1; // A0 ENTREE ADC 5REFERENCE
  //TRISC=0;
  TRISD=0;
  //////inintialisation des paramètres///
m1 = 0;
e1 = 0;
e2 = 0;
Lcd_Init();                        // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);             // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);        // Cursor off
  IntToStr(i,txt),
  Lcd_Out(1,1,"i=");               // Write text in first row
  Lcd_Out_Cp(txt);
while(1){
}
}
```

III.5.8.2.le schéma finale

On crée le schéma final qui contient toutes les composants électroniques qui sont conduisent par notre programme finale.

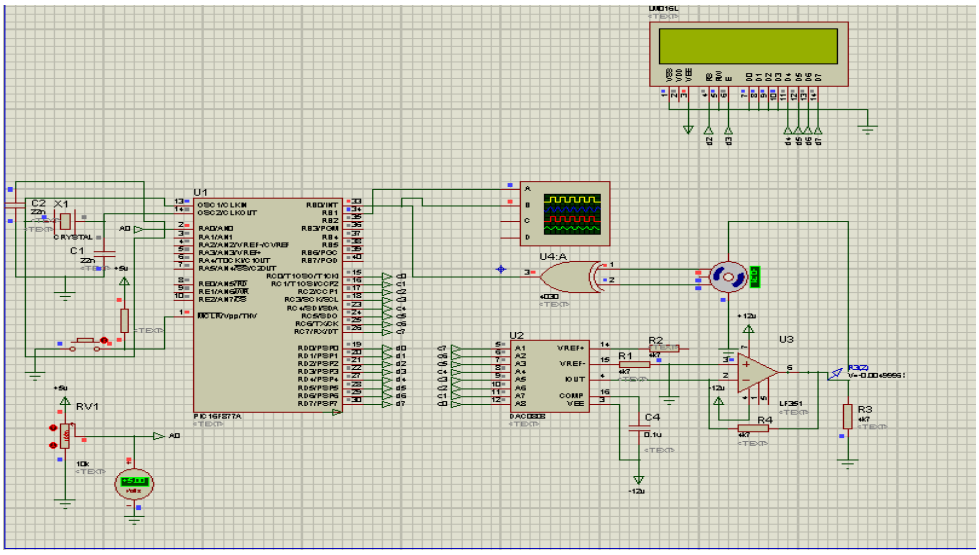
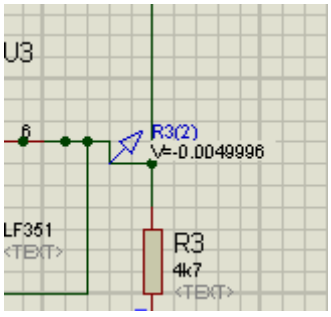
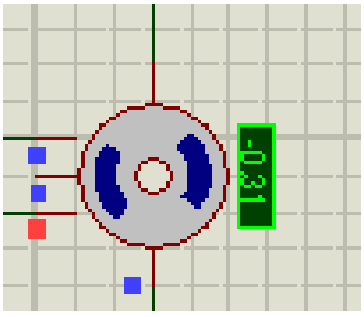


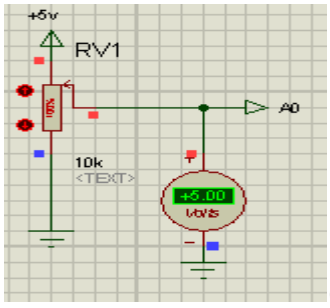
Figure III.29.*test de simulation pour Vref=1.25v.*



(a)



(b)



(c)

a: la tension de sortie au DAC

b: vitesse de la moteur consigné

c: la tension de l'entrée

III.5.8.3.Remarque

on constaté que la tension de sortie est bien chuté et n'est pas desiré car pour 5v il exige de 3000 RPM .

a cause de ça on essaye un solution altérnative pour éviter cet chute et pour conduire le moteur parfaitment .

cet solution s'appelle PWM(pulse width modulation) ou MLI (modulation largeur implusion).

a l'aide d'un transistor tip122 .

le solution de PWM vient pour remplacer le premier essaie .

et pour eviter l'aggrandisment et la large de la carte de commande par ajouter des nouveaux composants qui sont mal et créér des problèmes comme l'étage de puissance

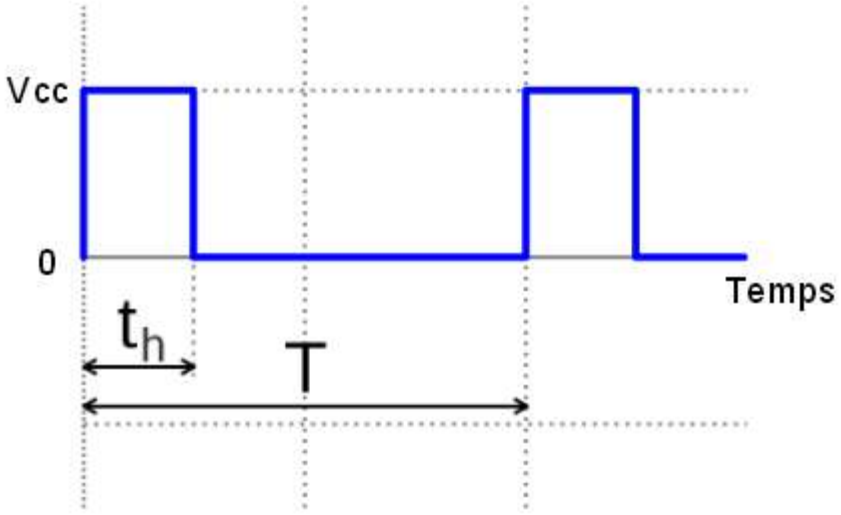
donc la solution de PWM vient pour minimiser les erreurs de calculs et n'est besoin pas des composants éléctrique assisté .

IV.5.9. Modulation de largeur d'impulsion (PWM)

Quand on veut faire varier la vitesse d'un moteur, la première idée qui vient à l'esprit est de faire varier la tension aux bornes du moteur mais on constate que pour des valeurs faibles de la tension, le moteur ne tourne pas. Le moteur demande une tension minimale pour démarrer. Si cette dernière est trop basse, les forces électromagnétiques ne sont pas suffisantes pour vaincre le frottement. Il devient donc difficile d'ajuster la vitesse de façon précise. La solution à ce problème est astucieuse. Il suffit de fournir au moteur une tension qui est toujours
la même soit la tension maximale ! Par contre, cette tension ne sera appliquée que par très courtes périodes de temps. En ajustant la longueur de ces périodes de temps, on arrive à faire tourner plus ou moins vite les moteurs. Mieux, on remarque que la vitesse des moteurs devient proportionnelle à la longueur des périodes de temps. Contrôler la longueur des périodes passées
à la tension maximale par rapport au temps passé sans application de tension (tension nulle) est donc le cœur de la solution. En réalité, cette solution est très connue en contrôle des systèmes et en électronique et elle porte le nom de PWM **(**Pulse Width Modulation**)** ou Modulation par Largeur d'Impulsions (MLI**)**.

III.5.9. 1.DÉFINITION

Le PWM est un signal numérique, donc la tension peut prendre deux valeurs seulement. Dans certains cas très spécifiques (onduleurs à MLI par exemple) on fabrique un troisième niveau en inversant la tension du niveau haut. Le signal est carré. Le niveau bas correspond généralement à 0 Volt. La période est notée T; la durée de l'impulsion (pour laquelle la tension est celle de l'état haut) est appelée th



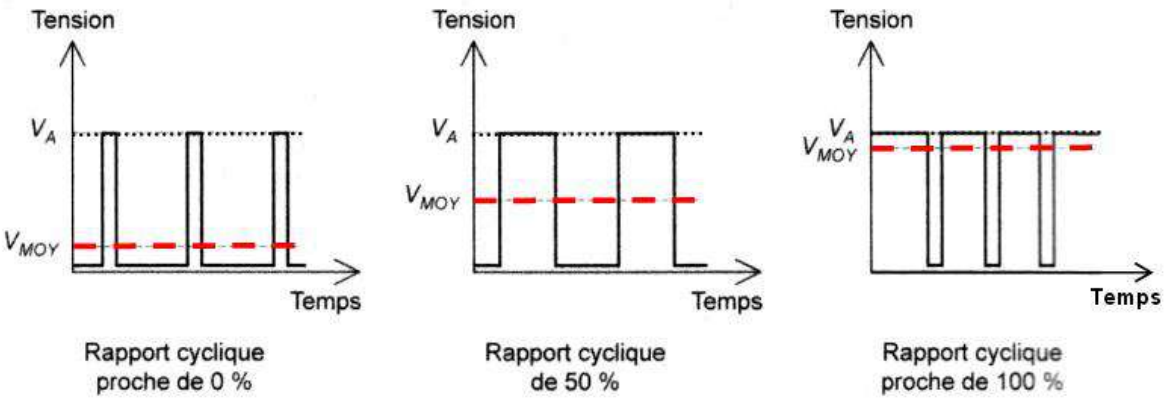
Si la période change, le signal n'est plus vraiment périodique au sens strict. On appelle alors T le pseudo période. [17]

Le principal intérêt de la technique PWM est de limiter la chauffe des composants électroniques. Par exemple : une lampe de 20 Watts allumée au maximum consomme 20 W. Si par une commande de gradation elle est allumée au quart de sa puissance, elle consomme 5 W. Le composant analogique devrait alors dissiper 15 W, ce qui implique un énorme radiateur. En PWM, la puissance fournie est soit maximale, soit nulle. Lorsqu'elle est maximale, pendant un quart du temps par exemple, il n'y a pas besoin de dissiper de puissance résiduelle. Lorsqu'elle est nulle, il n'y a pas besoin de dissiper non plus de puissance, car elle n'est pas fournie du tout. Un autre intérêt du PWM est que la tension appliqué
e au moteur pendant thés Vcc. Celle-ci est suffisante pour vaincre les frottements et faire tourner le moteur. La tension moyenne appliquée au moteur est proportionnelle au rapport cyclique, ce qui permet d'avoir des consignes de vitesse faibles :

$$V_{MOY}=t_h \times Vcc/T$$

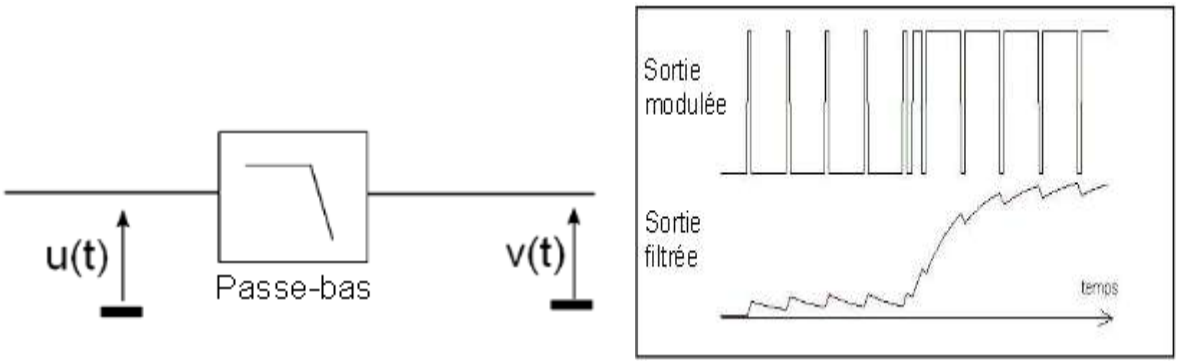
III.5.9. 2.FRÉQUENCE

La commande d'actionneurs de puissance par PWM est très liée à la notion de fréquence. Pour que l'impression d'une valeur moyenne constante d'allumage apparaisse, il faut que l'alternance d'allumage/extinction soit suffisamment rapide pour qu'elle ne se remarque pas. Selon les utilisations la fréquence du PWM va de 100 Hz (100 cycles par seconde) à 200 kHz.



III.5.9. 3.Remarques

Même si le signal n'est nullement une tension analogique, il peut le devenir si on lui fait subir un filtrage de type passe-bas. Filtrer ce signal carré permet d'obtenir la tension moyenne. La fréquence de coupure du filtre doit être plus faible que la fréquence du signal carré. Autrement dit la fréquence porteuse doit être significativement supérieure à la fréquence de coupure du filtre, pour que l'ondulation résiduelle soit faible. . [17]



III.5.9. 4.RAPPORT CYCLIQUE

On appelle rapport cyclique le rapport :

A =100 x t h /T

Exprimé en pourcentage.

Si th =0 alors α=0% et la tension moyenne de sortie

Est nulle.

Si th=T alors α=100% et la tension moyenne de

Sortie est égale à Vcc. . [17]

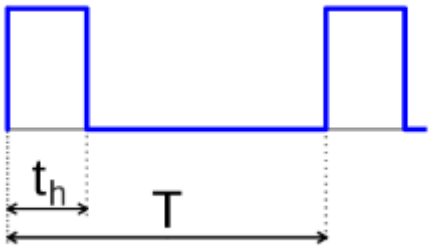


Figure III.30.*rapport cyclique*.

IV.5.9.5. Program PWM sur MIKROC

```
// Define LCD module connections.
 sbit LCD_RS at RB2_bit;
sbit LCD_EN at RB3_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;
sbit LCD_RS_Direction at TRISB2_bit;
sbit LCD_EN_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;
// End LCD module connection definition
 float duty;
 unsigned int v;
 char str [16];
 char st [16];
 float rpm ;
```

```
void main()
{
 TRISB.B1=0;
 PORTB.B1=1;
 Lcd_Init();        // Initialize LCD
 ADC_Init();
 Lcd_Cmd(_LCD_CLEAR);             // CLEAR display
 Lcd_Cmd(_LCD_CURSOR_OFF);        // Cursor off
  for(;;)
        {
         PWM1_Init(10000);
         PWM1_start();
         v = ADC_Read(0);
         duty=(v)/4;
         PWM1_Set_Duty(duty);
         floattostr(duty,str);
         lcd_out(1,1,"duty=");
         lcd_out_cp(str);
         TMR0=0;
         delay_ms(10);
         rpm=TMR0*60;
         floattostr(rpm,st); delay_ms(10);
         lcd_out(2,1,"rpm=");
         lcd_out_cp(st);
        }
 }
```

Dans ce cas, nous devons relier le transistor TP 122 avec moteur comme schéma ci-dessous:

Proteus

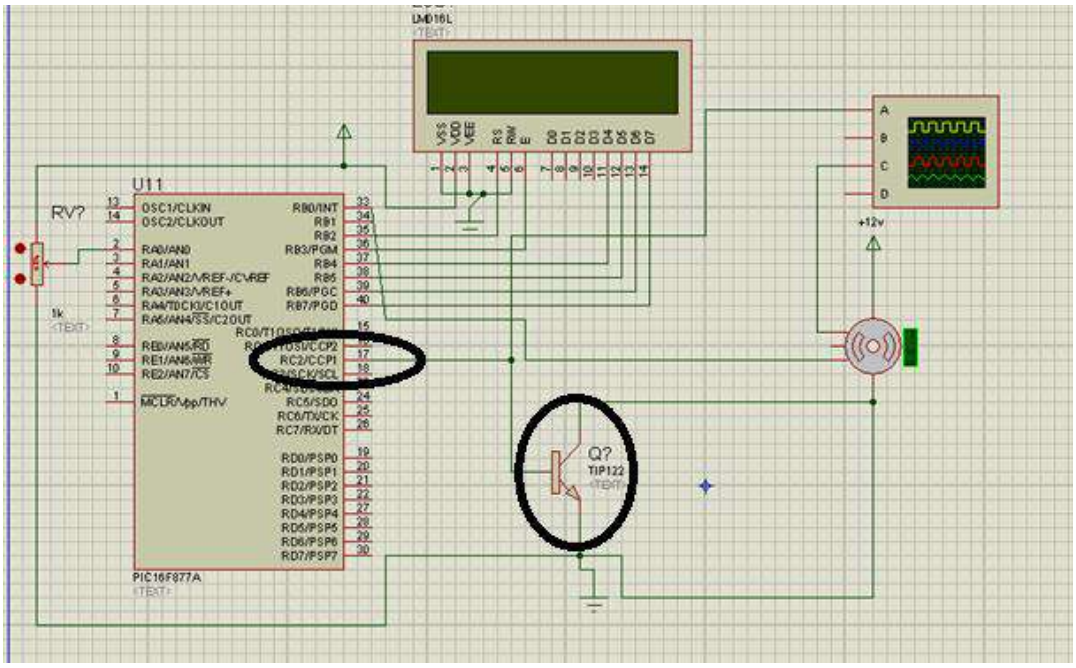


Figure III.31.*Schéma globale de PWM sur le proteus.*

III.5.9.6. SIMULATION ET ESSAIE PRATIQUE *PWM*

1ere essaie confirme large de 255 est donne 2900 RPM et Résultats sur l'oscilloscope de large de 255 bits

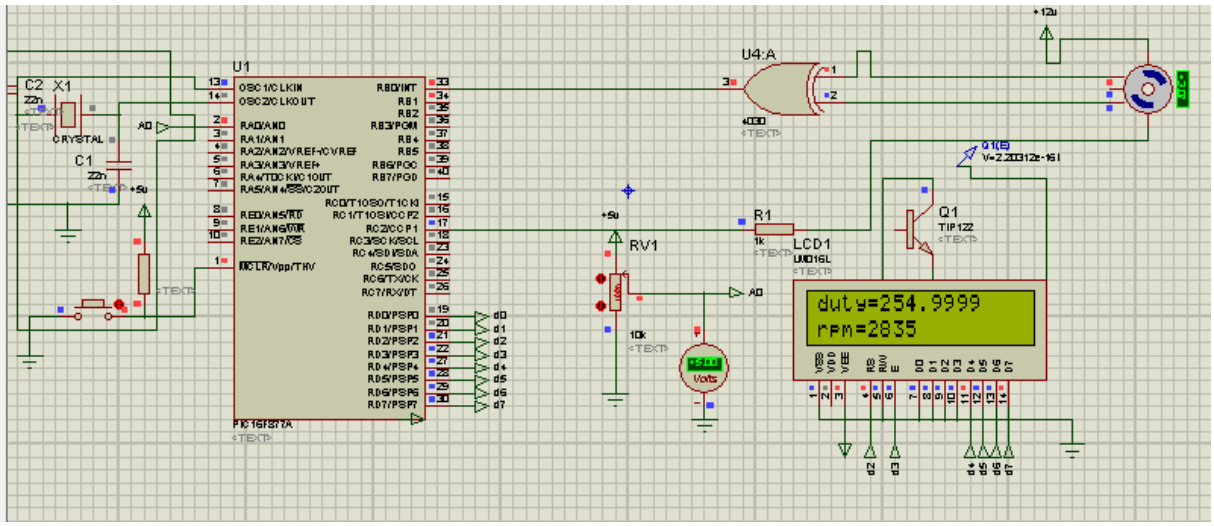


Figure III.32.*Schéma de la simulation pour* PWM 255bits

(RPM =2835tr/min).

2eme essaie :PWM en large 128 et résultats sur l'oscilloscope. PWM en 128 bits

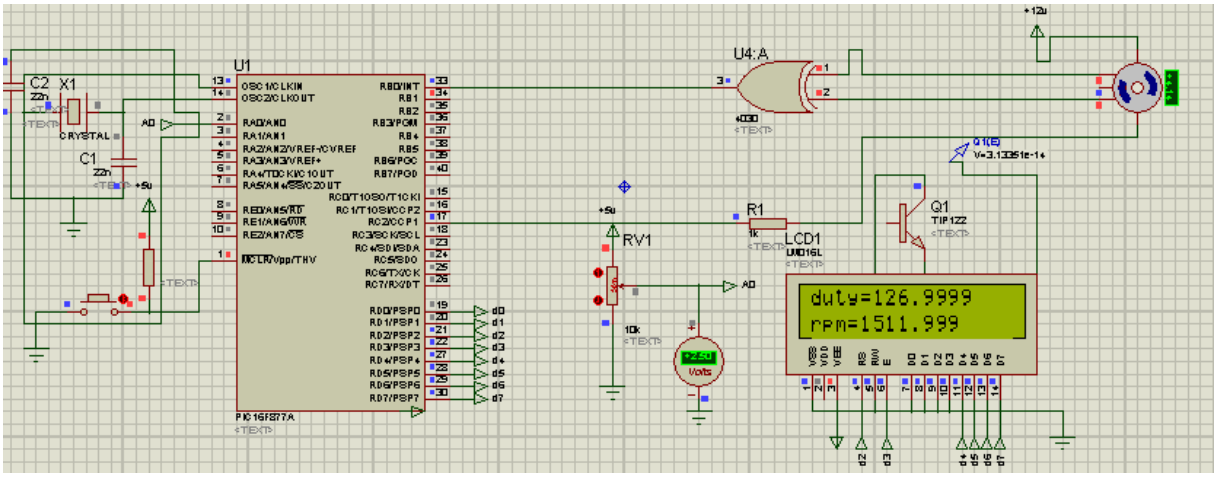


Figure III.33.*test de la simulation pour* PWM 128 bits

(RPM =1511r/min).

3eme essaie de PWM en 64bits et Résultas sur l'oscilloscope de PWM 64 bits

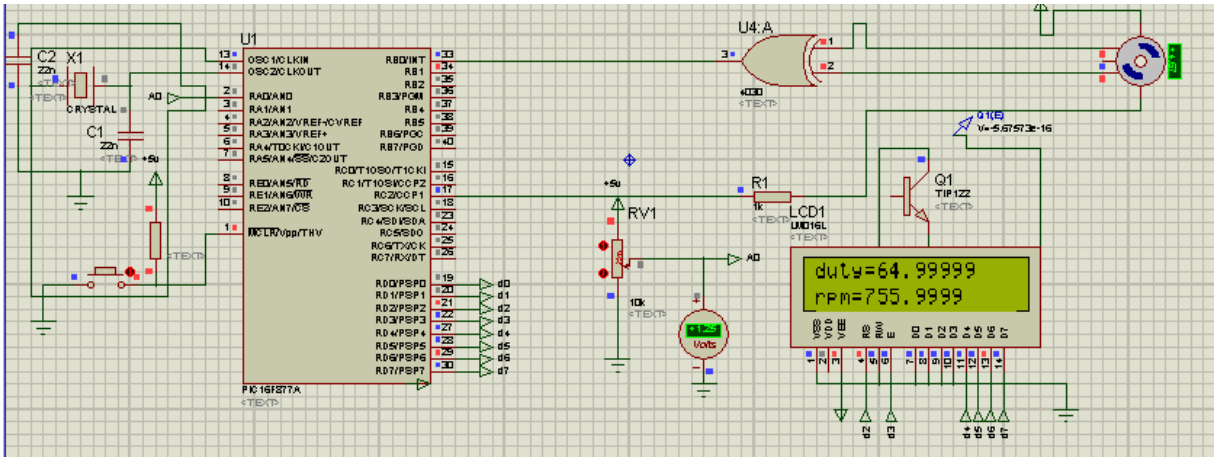


Figure III.34.*test de la simulation pour* PWM 64 bits

(RPM =755tr/min).

III.5.9.7.Essai pratique PWM

Au première essaie on a réalisé le PWM.

1ere essaie confirme large de 255 est donne 2900 RPM et Résultats sur l'oscilloscope de large de 255 bits

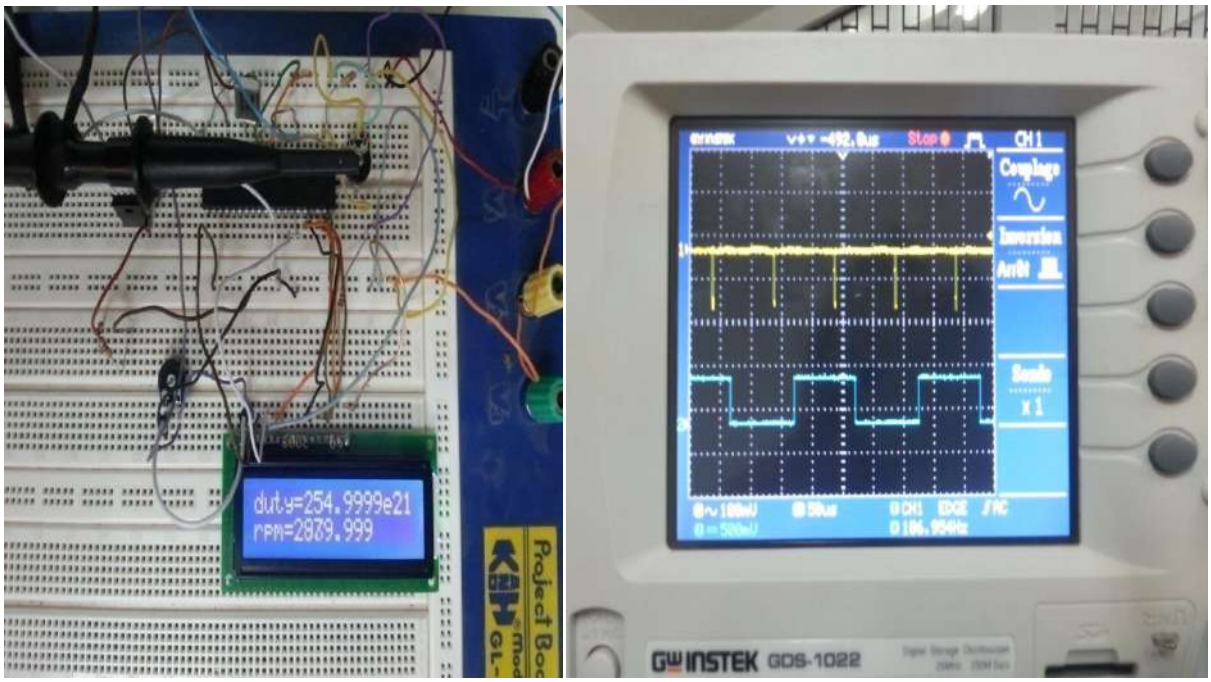


Figure III.35.*Test de PWM pour 256 bits.*

2eme essaie :PWM en large 128 et résultats sur l'oscilloscope. PWM en 128 bits

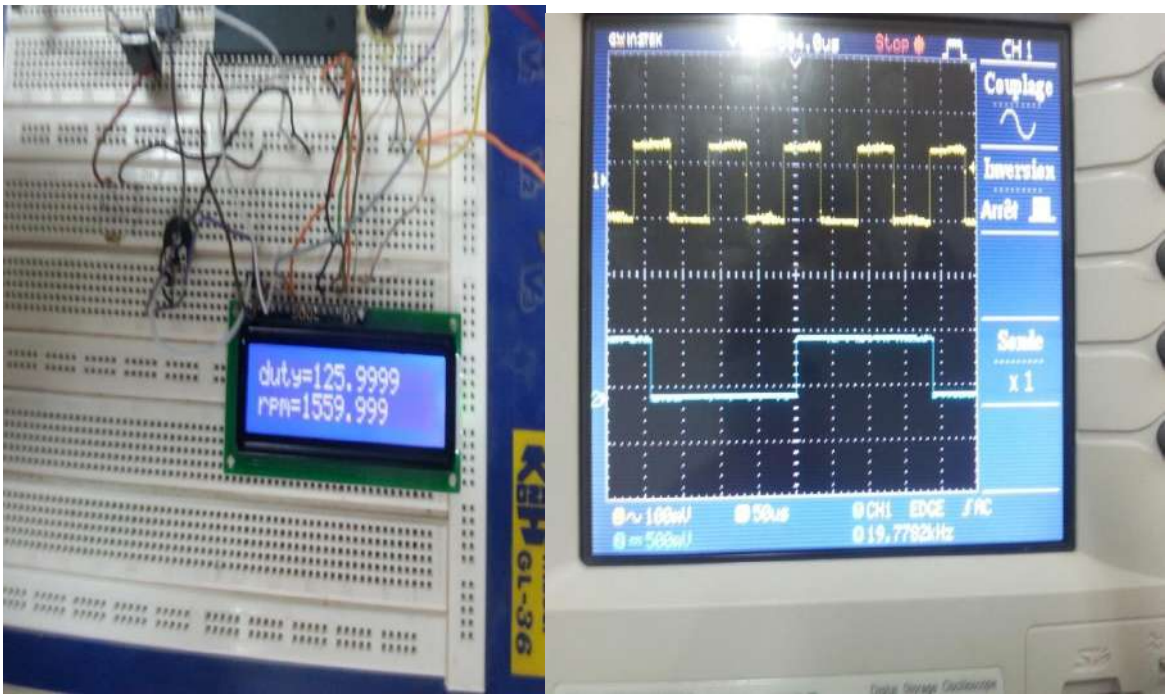


Figure III.36. Test de PWM pour 125bits.

3eme essaie de PWM en 64bits et Résultas sur l'oscilloscope de PWM 64 bits

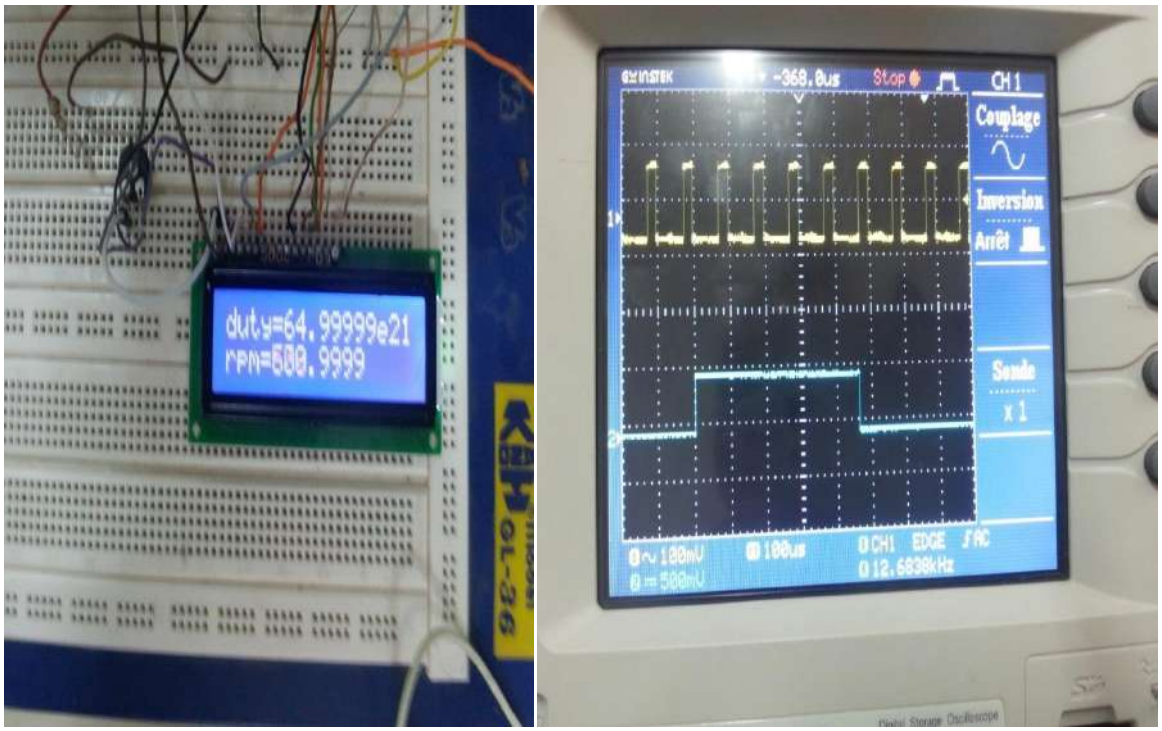


Figure III.37. *Test de PWM pour 64 bits.*

III.*5.9.8.* Résultat finale basé sur la PWM

d'aprés les essaies de conduire le moteur avec le PWM et le circuit avancé (dac+ampliope) on trouve une chute de tension au niveau de DAC .

pour ça on remplaçant le circuit par une autres methode (PWM°) pour eviter largement des composants (étage de puissance) Nous devons utiliser transistor.

donc le simulation finale consiste la régulation PI + PWM .

III .5.9.10..Programme finale MIKROC

```
///////////////déclaration des variables//////////////////
  char txt[14],txt1[14]  ;


  float T=0.01 , TPM_ref,TPM,ref,U;
  int  counter,compt;
 // float e=0.0,pe,ie=0.0,de,iep=0.0;
```

```
  float m0,m1,e2,e1,e0,Q0,Q1;
 unsigned  short duty;
 float Kp=0.62,Ki=84.57,Kd=0.0;
////////////////////////////////////////////////////////
// programme d'interruption//////
void interrupt() {
if (INTCON.INTF)
  {
   counter++;          //bascule de l'état
   INTCON.INTF = 0;     // mise a zero le drapeau de l'interruption de rb0
  }
else if (PIR1.TMR1IF)
 {
 PORTB.F3=PIR1.TMR1IF;
 TMR1H        = 0xB1;
 TMR1L        = 0xE0;
 ////////////////////calcule de la vitesse//////////////////
 compt=counter;
 TPM= (60/(200*T))*compt;    // nombre de tour par minute  T=10ms; rev=200
 counter=0;           // réinitialisation du counter
 PIR1.TMR1IF=0;
 PORTB.F3=PIR1.TMR1IF;
}
 }
void main(){
////////////////////////////////////////////////////
 Lcd_Init(&PORTD);        // Initialize LCD connected to PORTD
```

```
 Lcd_Cmd(Lcd_CLEAR);       // Clear display
 Lcd_Cmd(Lcd_CURSOR_OFF);  // Turn cursor off
 ////////////////////////////////////////////////////////
INTCON.GIE= 1;        // ACTIVATION DES INTERRUPUTION GLOBALE
INTCON.PEIE=1;        //ACTIVATION DES INTERRUPTION DES PERIPHERIQUES
INTCON.INTE=1;        //  ACTIVATION DE L'NTERRUPTION  int_rb0
PIE1.TMR1IE=1;      // ACTIVATION L'INTERRUPTION DU TIMER1
INTCON.INTF = 0;   // mise a zero le drapeau de l'interruption de rb0
PIR1.TMR1IF=0;     // mise a zero le drapeau de  du timer1
ADCON1= 142; /*AN0,AN1 entrées analogique   /*  AN2=Vss ;AN3=Vdd  */
  T1CON         = 0x01;    // T=10 ms
  TMR1H         = 0xB1;
  TMR1L         = 0xE0;
 TRISB=0b00000001;              // Rb0 entrée dd l'encodeur
 TRISA.F0=1;                   //  A0 ENTREE ADC 5REFERENCE
 TRISC.F2 = 0;                 //  Sortie de pwm
 PWM1_Init(10000);             //   Initialisation
 //////inintialisation des paramètres////
Q0 = 1.2;     //Kp + Ki*T/2.  ;
Q1 = 1.2;     //-Kp + Ki*T/2. ;
m1 = 0;
e1 = 0;
e2 = 0;
while(1){
//delay_ms(100);
   /////////////// Lecture de la référence ////////////////
    ref = ADC_Read(0);      //lecture de la valeur  référence
```

```
   U=1.46*ref;      // converession tension--vitesse  3000./1023=1.46~~1.5
   m1 = m0;
   e1 = e0;
   e2=e1;
   e0= U -TPM;
   m0 = m1+ Q0*e0+Q1*e1 ;     //0.18  -0.1   T=10ms

   if (m0>255){
   m0=255;
   }
 if (m0>255) m0=255;       // Limitation
   ////////////////// fin de régulation ////////////////////
   PWM1_Start();
   duty=m0;                  //Affectation
   PWM1_change_Duty(m0);   //sortie de la commande sous forme de PWM
  intToStr(U,txt);
  Lcd_Out(1,1,"TRM_ref=");                // Write text in first row
  Lcd_Out_Cp(txt);
  IntToStr(TPM,txt1);
  Lcd_Out(2,1,"TPM =");                // Write text in first row
  Lcd_Out_Cp(txt1);
}
}
```

III .5.10. résultat de simulation

Pour Kp=4.77et Ki=39.27

Nous avons pris la valeur de

1. On prend la valeur de l'alimentation V_{ref} = 1.25Vet ***(RPM*** **=3000 tr/min)**

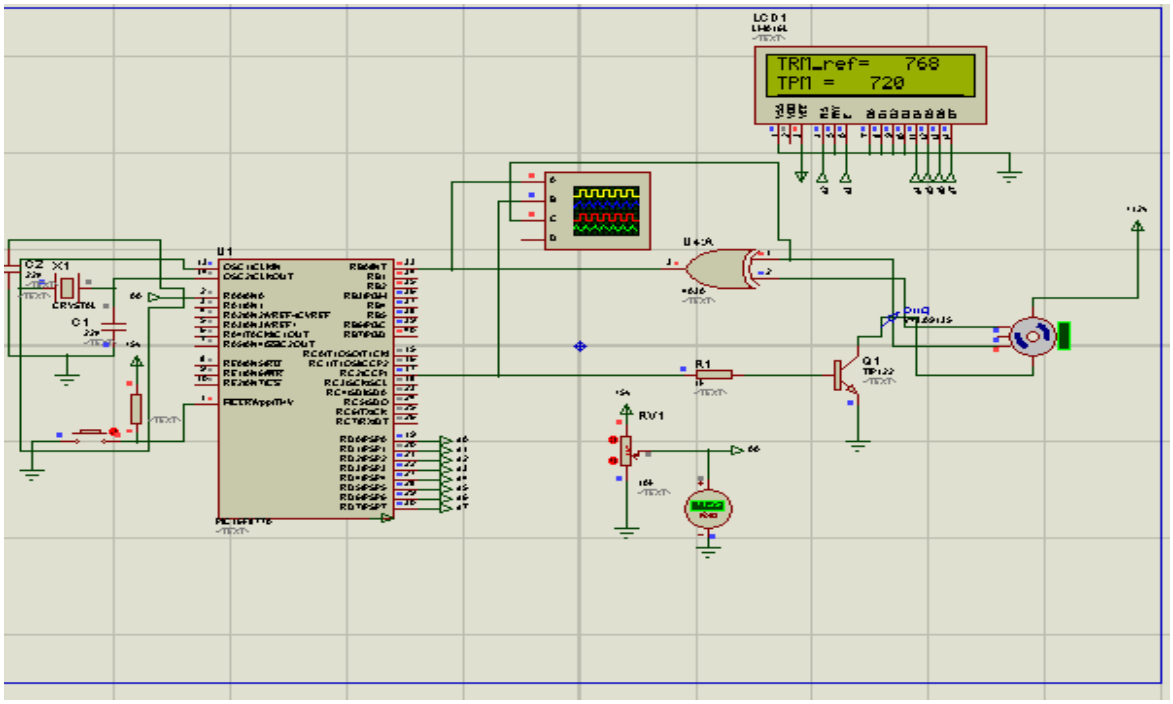
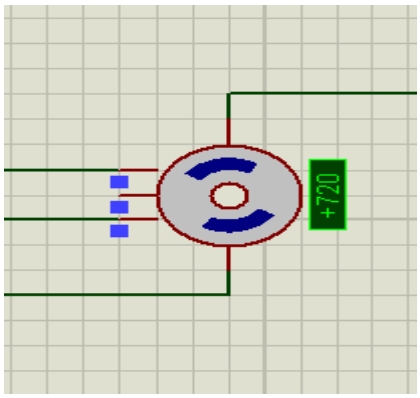
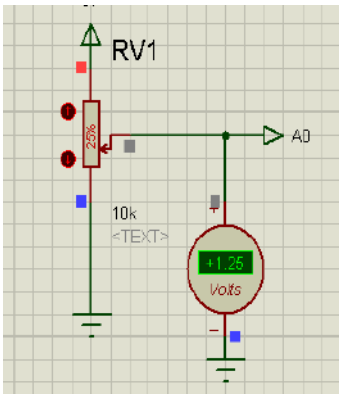
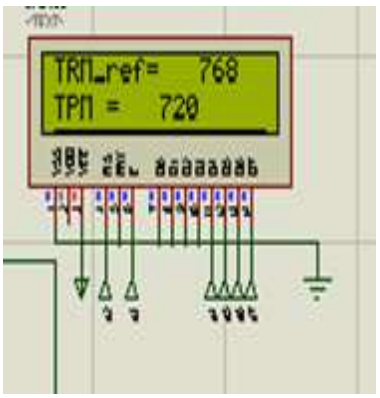


Figure III.38. *Test de simulation pour1.25 v.*







a)RPM B) tension REF c)Affichage LCD

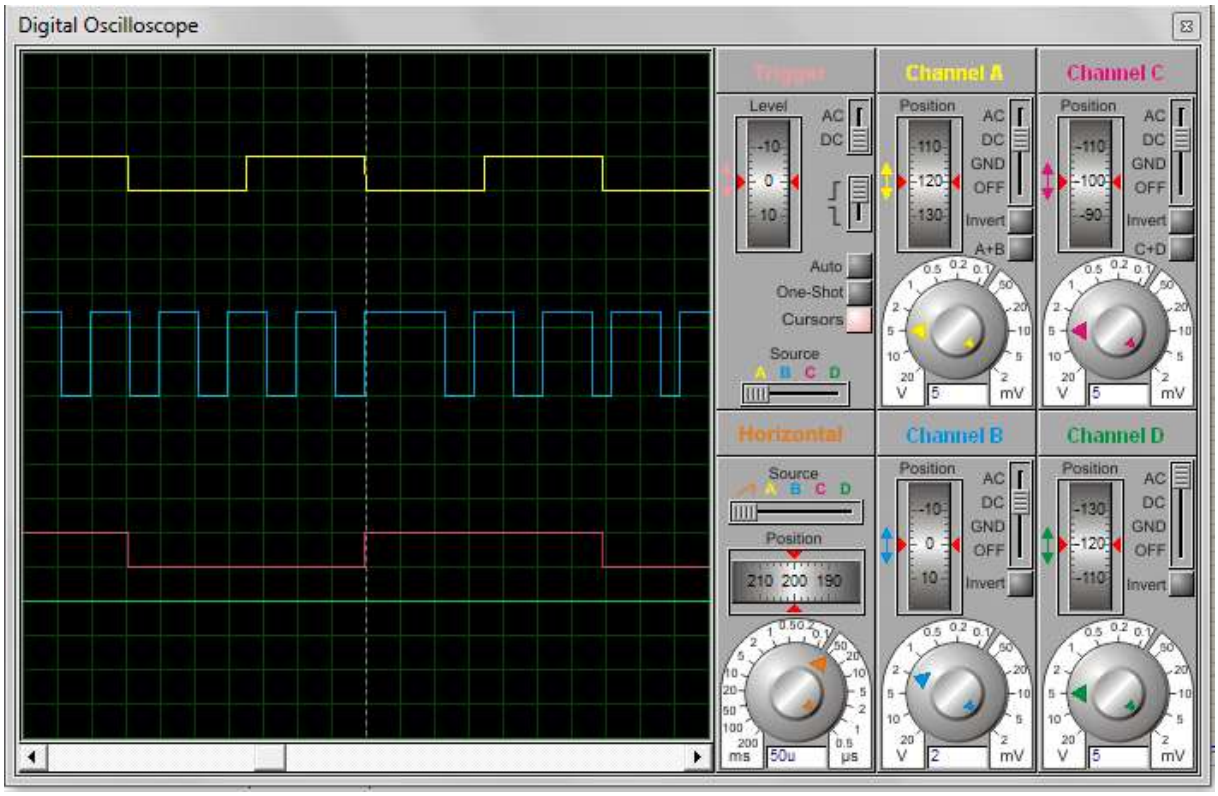


Figure III.39. *L'affichage sur l'oscilloscope.*

2. On prend la valeur de l'alimentation V_{ref} = 2.5V et ***(RPM*** =3000 **tr/min)**

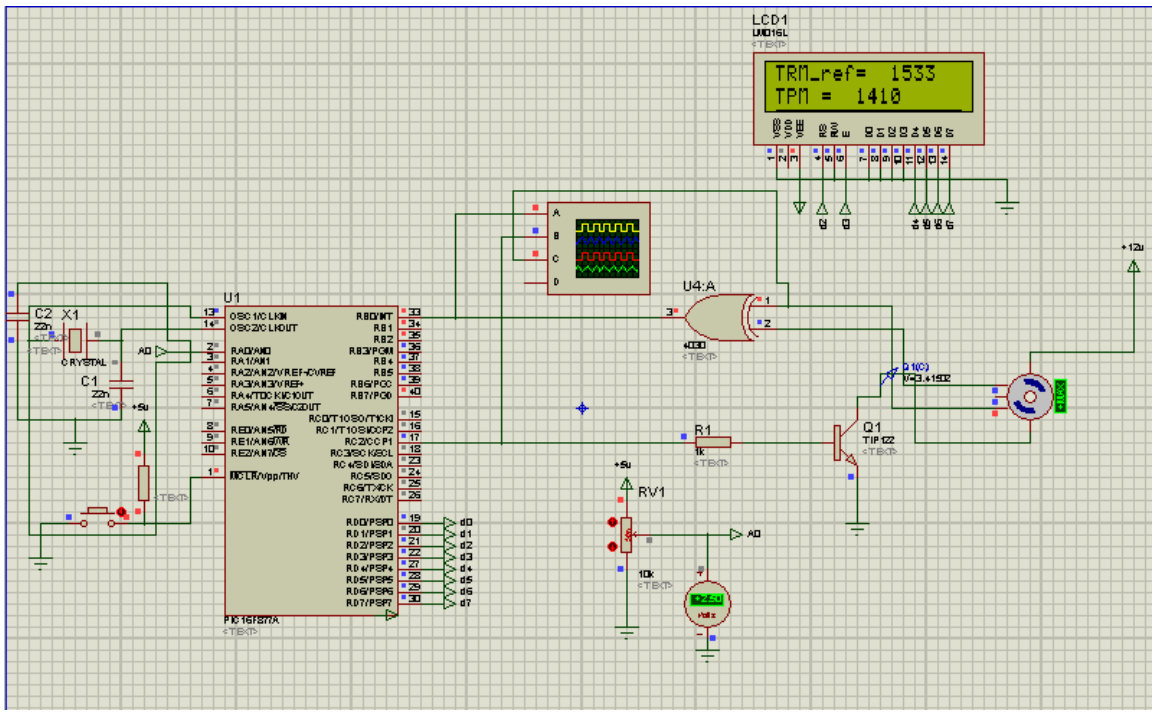
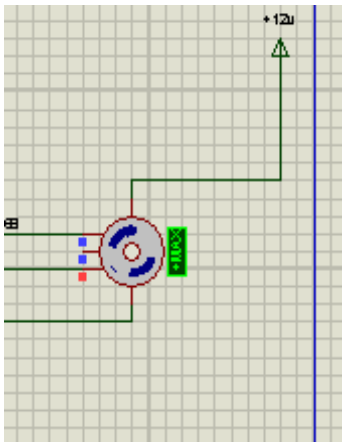
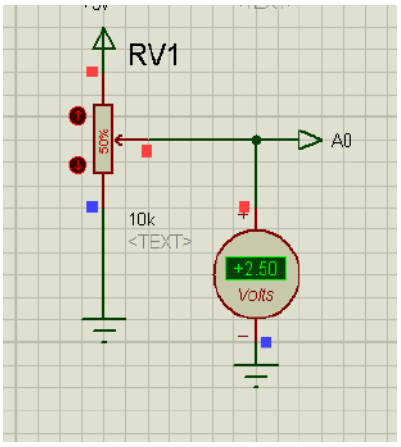
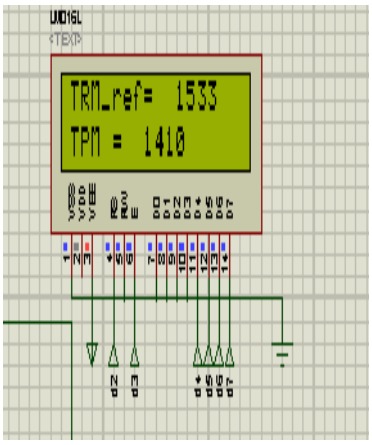


Figure III.40.*test de la simulation pour 2.5v.*







(d)RPM (e)tension REF (f) Affichage LCD

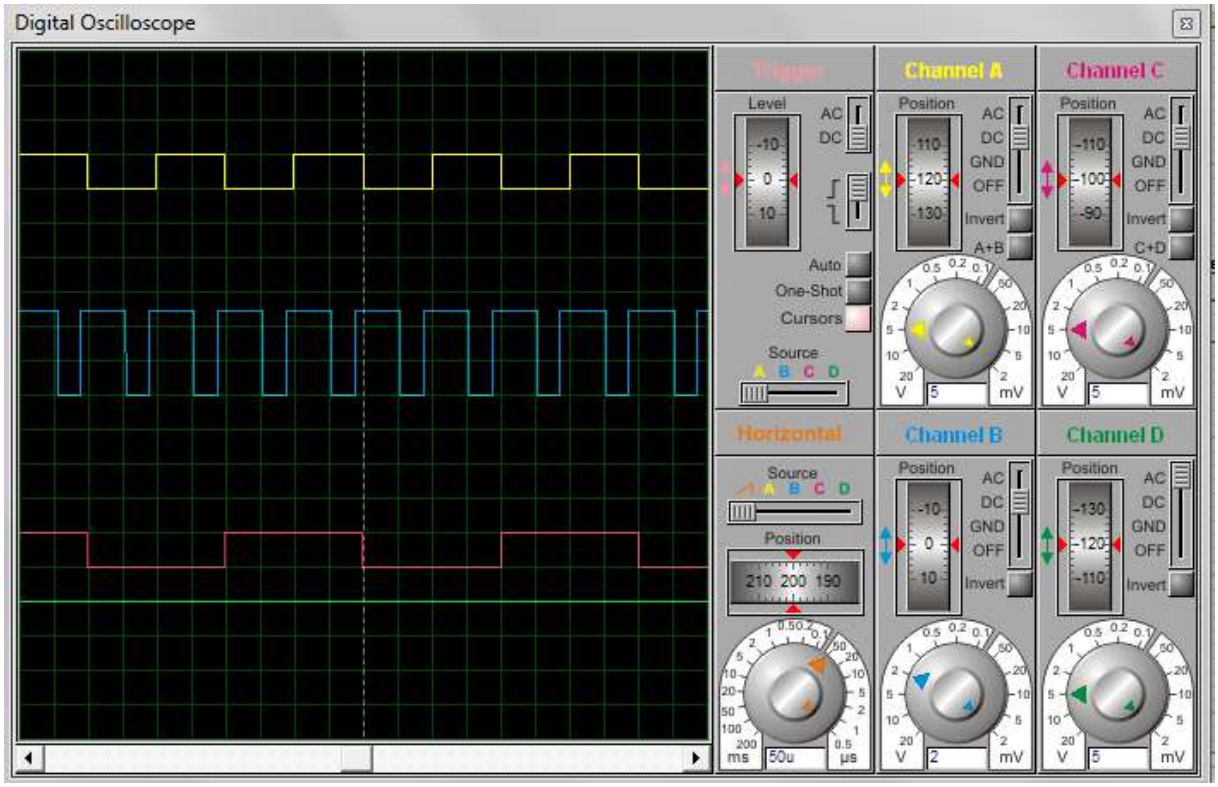


Figure III.41.*l'affichage sur l'oscilloscope.*

3. On prend la valeur maximal de l'alimentation V_{ref} = 5V et la valeur de rpm =3000 **tr/min**

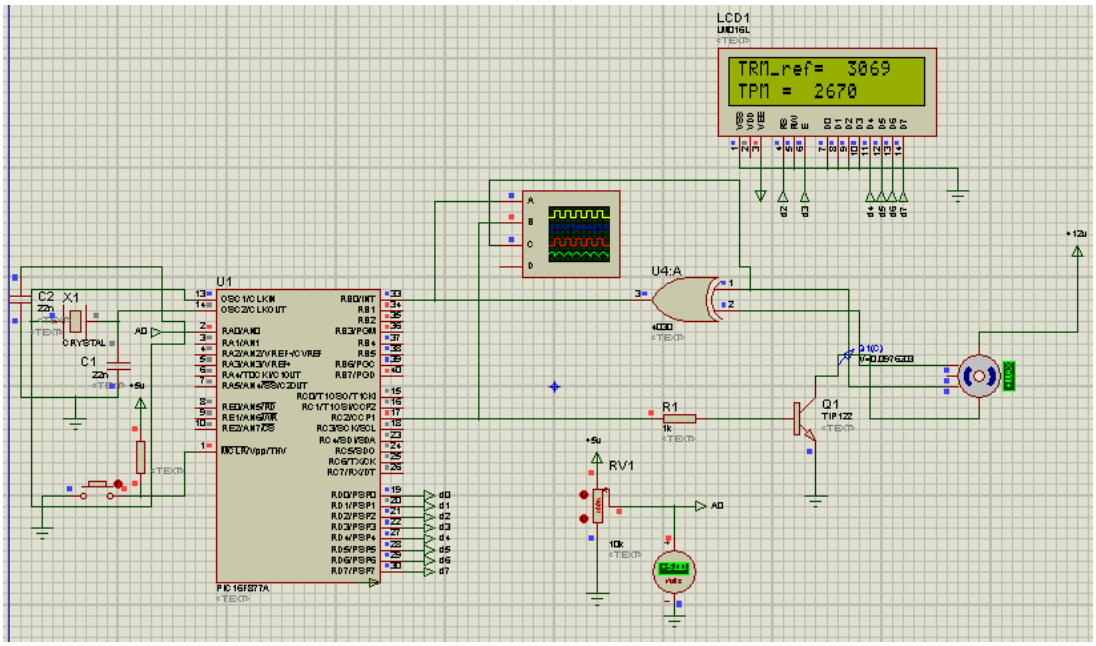
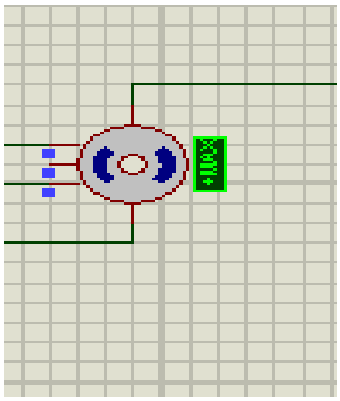
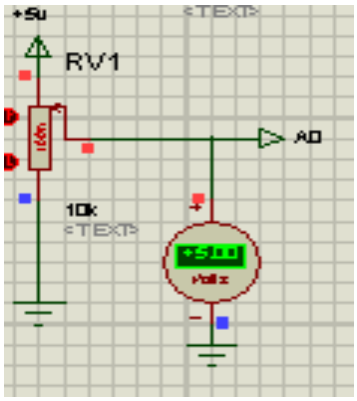


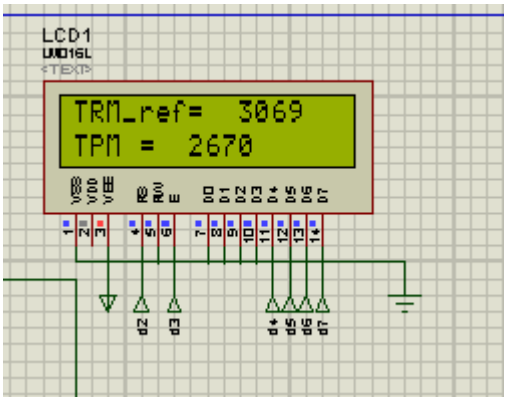
Figure III.42.*test de simulation pour 5v.*



(g)RPM



(h) tension REF



(i) Affichage LCD

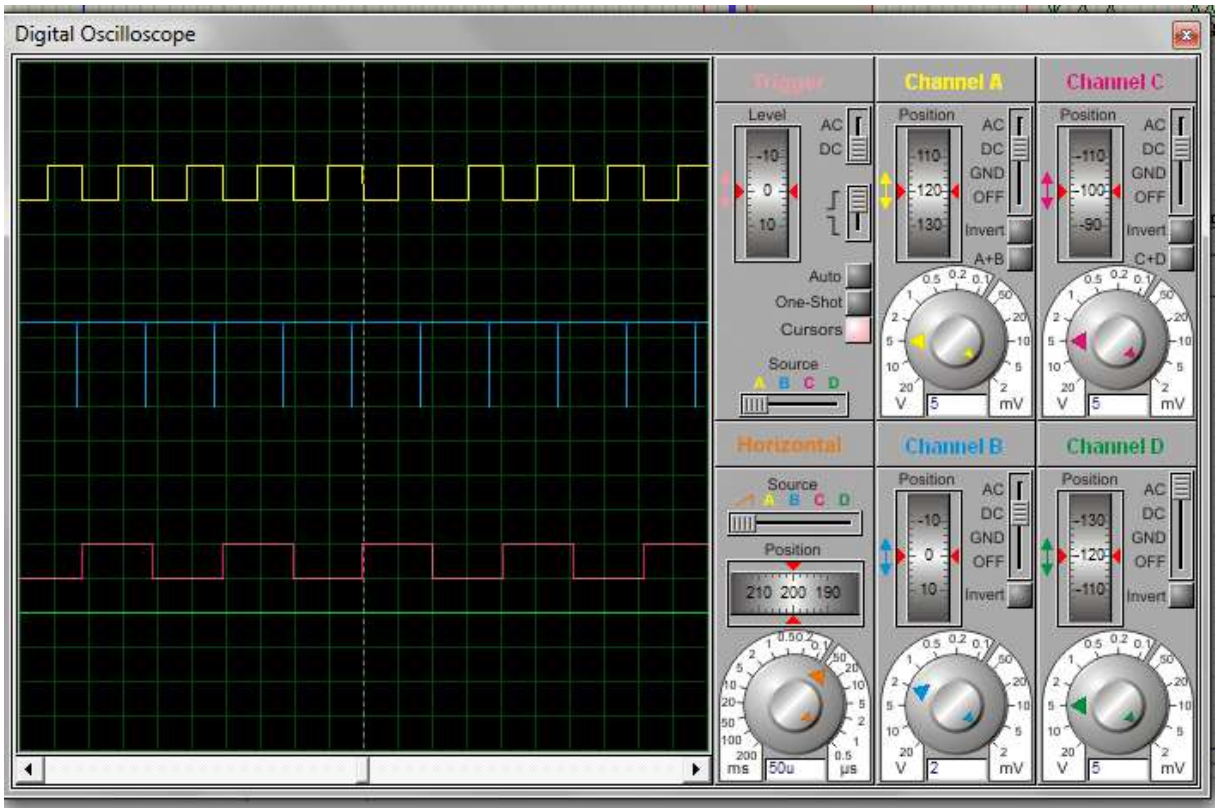


Figure III.43.*l'affichage sur l'oscilloscope.*

IV.5.11. réalisation final

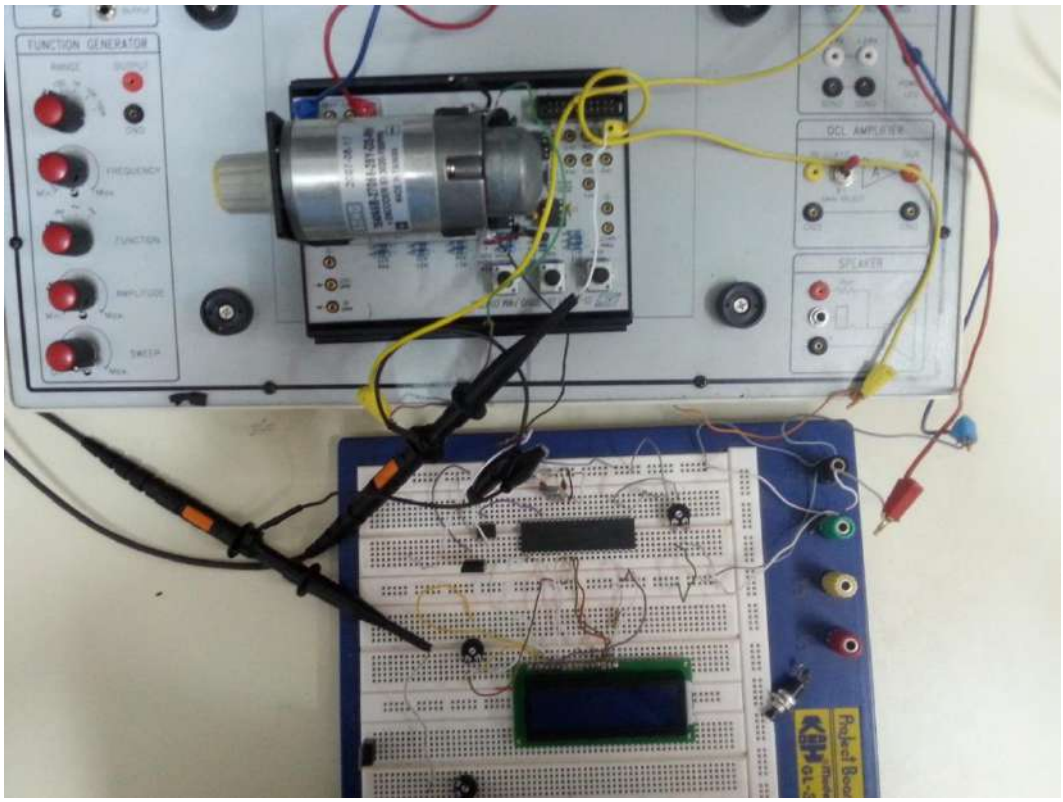


Figure III.44. *Schéma de réalisation final.*

1eme essaie de réalisation pour v ref =1.25 v est rpm=3000 tr/ min sur plaque d'essais

Nous avons obtenu les résultats suivants sur LCD et oscilloscope

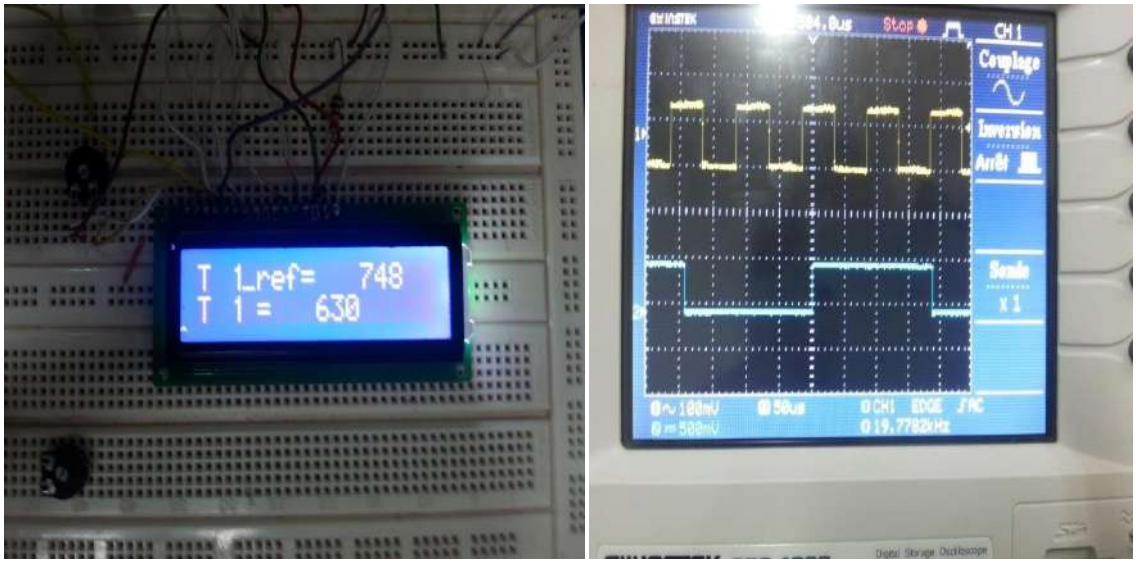


Figure III.45.*test de réalisation et affichage oscilloscope pour Vref=1.25v***.**

2eme essaie de réalisation pour Vref =2.5 v est rpm=3000 tr/ min sur plaque d'essais Nous avons obtenu les résultats suivants sur LCD et oscilloscope

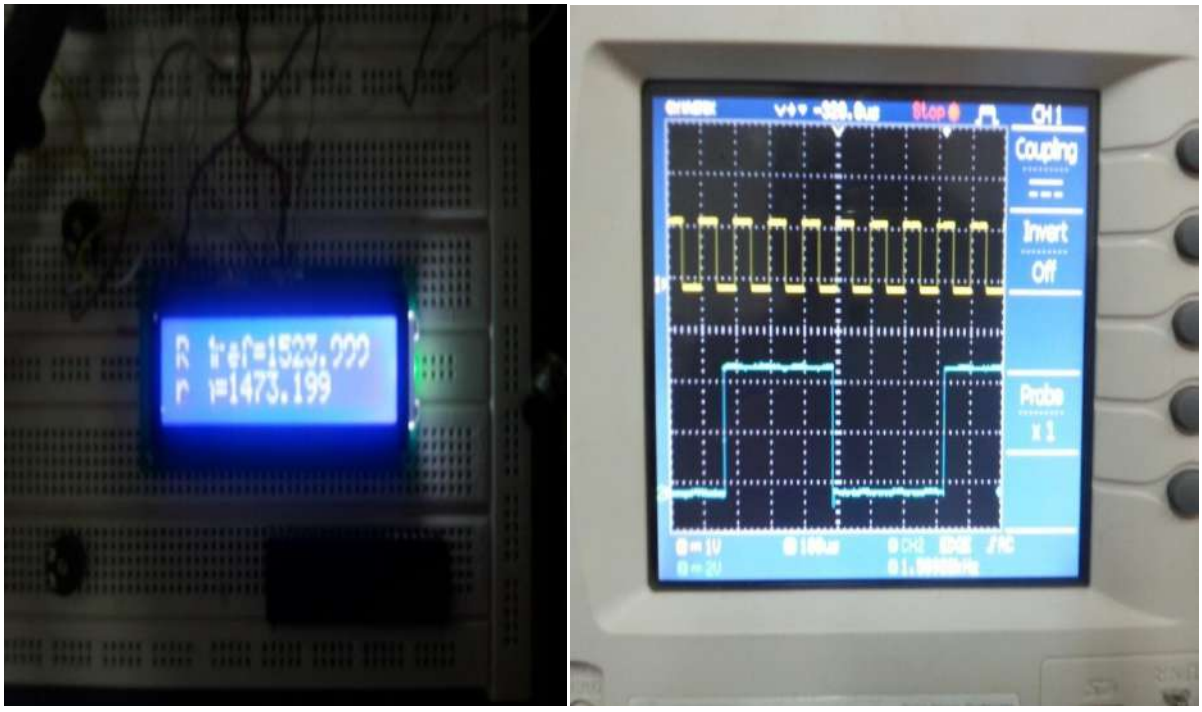


Figure III.46*.test de réalisation et affichage oscilloscope pour Vref=2.5v.*

3eme essaie de réalisation pour Vref =5 v est rpm=3000 tr/ min sur plaque d'essais Nous avons obtenu les résultats suivants sur LCD et oscilloscope.

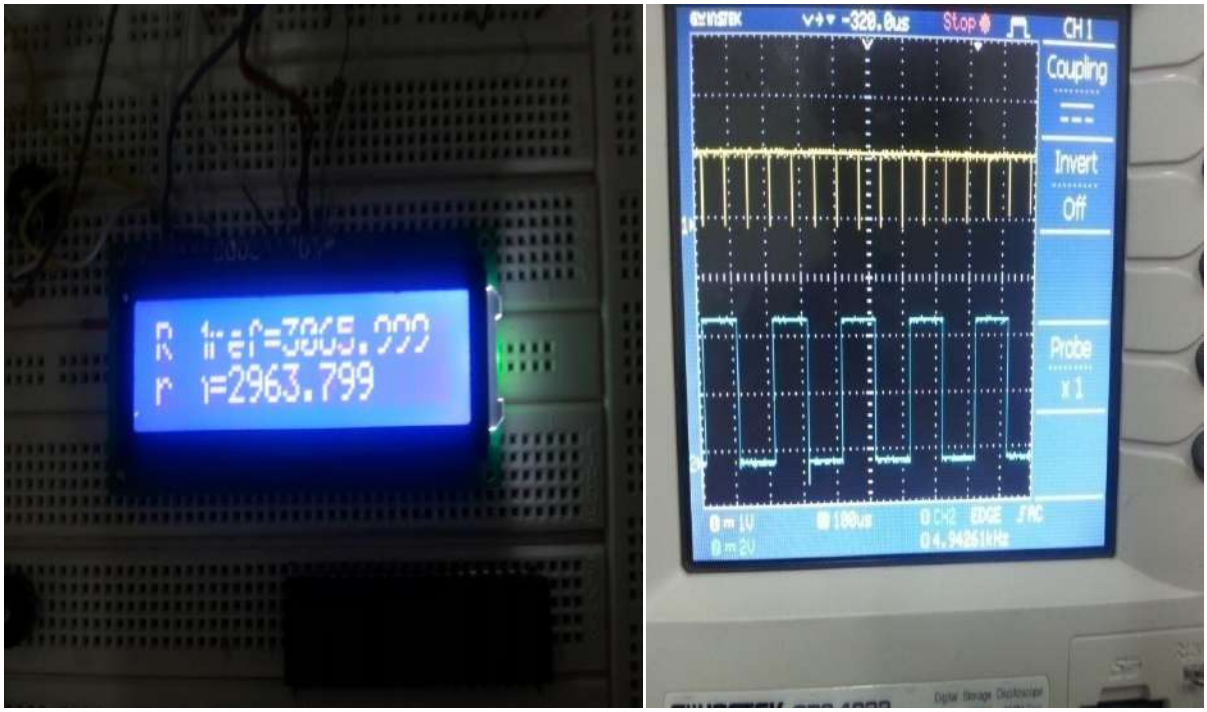


Figure III.47. Test de réalisation et affichage oscilloscope pour Vref=5v**.**

III.12. CONCLUSION

Dans ce chapitre nous avons réalisé toutes les taches citées précédemment qui sont concrétisé par le programme proteus ISIS.

On a configuré deux méthodes pour commander la vitesse de MCC.

La première c'est compliqué par le DAC qui exige un circuit large et long qui on a un problème de l'étage de puissance.

La deuxième c'est simple et efficace et n'exige rien .un parcours de commande simple et directe par l'outil de PWM qui nous permettre à varier la vitesse de la moteur par une façon simple et rapide et plus stable que la premier configuration.

Après varier la fréquence de PWM .on donnant 03 essaie pour voir la flexibilité de cette solution qui nous permettre à varier la vitesse de MCC parfaitement.

Le pic et le moteur sans aucune intermédiaire par la solution de PWM.

Bien sûr que la commande exige un correcteur ou régulateur qui corrige l'erreur entre la vitesse mesuré et la vitesse consigné.

Donc on implémente un régulateur PI qui fonctionne sur l'actionneur proportionnel et l'actionneur intégral.

Dans ce chapitre ; on réalisant la commande d'un moteur et on vécut des résultats approximés et n'est pas efficace à 100%.parceque on a des obstacles.

Conclusion générale

Ce travail à été pour nous, le premier pas vers un monde nouveau et complexe, mais très intéressant notre recherche bibliographique dans les domaines de l'électronique, de l'électrotechnique et de l'informatique tel que « les interruptions électronique, les machines à courant continu et la programmation », nous a permis d'enrichir nos connaissance et de développer notre base théorique.

Dans ce travail nous avons essayé d'apporter d'utiliser et d'employer les possibilités offertes par l'emploi des PICs dans le domaine des machines à courants continu. Notre objectif principal était donc de fournir à la fois un outil moins coûteux et d'une utilité.

- Nous avons implémenté le régulateur PI dans un microcontrôleur PIC pour la commande de la vitesse de la machine à courant continu.
- La régulation de la vitesse est réalisée à l'aide du PIC.

Toute fois le champ d'investigation est très large et les méthodes très nombreuses. Des améliorations peuvent être apportés à ce travail afin de le perfectionner et de l'enrichir.

_ Au niveau du circuit de commande.

_ Au niveau de la régulation.

Par exemple de compléter ce travail afin de commander et de réguler la vitesse d'un moteur dans les sens de rotation.

L'implémentation de l'algorithme PI par le microcontrôleur PIC16F877A a nécessité la maîtrise de l'architecture de ce dernier et une familiarisation avec son environnement de développement.

Le circuit PIC16F877A est très intéressant par son rapport qualité/prix.

Nous faisons une réalisation pratique nécessitant une commande d'une machine à courant continu en temps réels, afin d'avoir une grande précision.

C'est des empirements qui nous poussent à faire un circuit real de commande de MCC

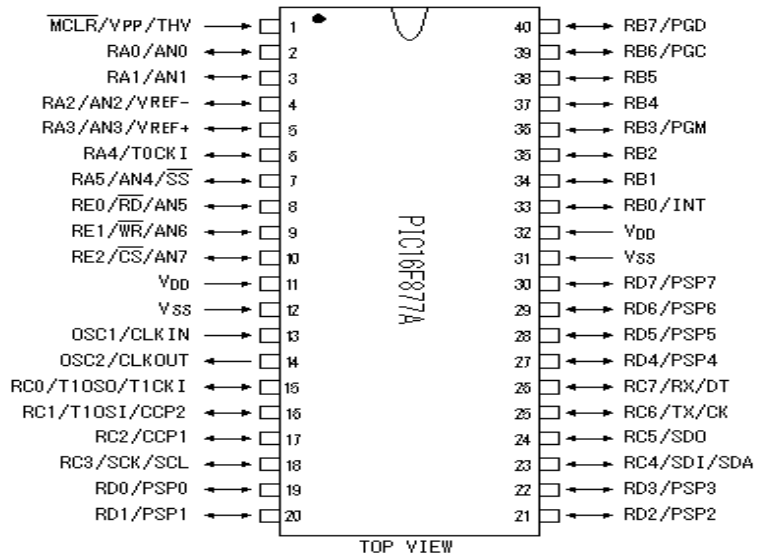
On finissait le travail par commander MCC par deux façons.

La première par le DAC mais on constate des inconvénients comme la chute de tension et l'agrandissement de notre circuit, la deuxième par PWM qu'il était la meilleure solution pour la commande.

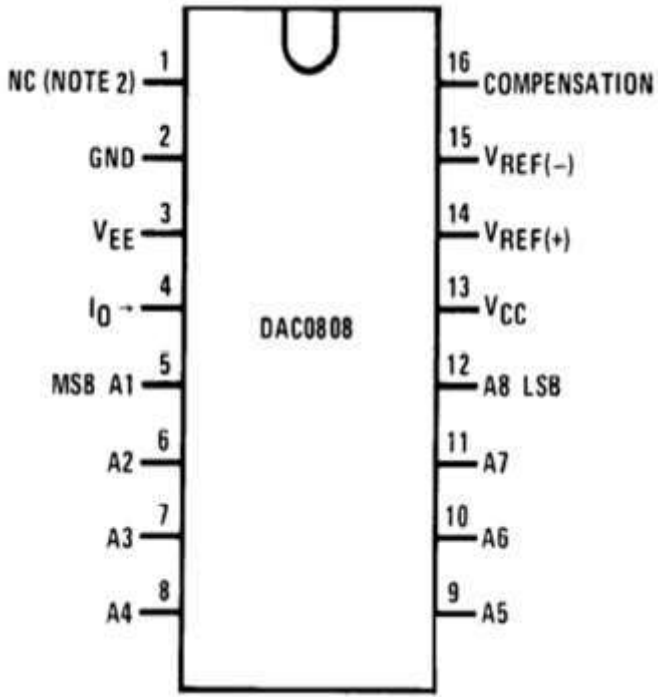
Donc notre travail comporte des idées approximé qu'on essaye toujours à l'améliorer.

Annexe

L'annexe 1 : DATACHEET de microcontrôleur pic 16f877a.



L'annexe 1 : DATACHEET de DAC0808



III.3. moyens de réalisation

1. Pic 16F877A



2. L'oscilloscope



3. Générateur d'impulsions :



4. servo-motor 27058-28Y-DS-RH-100R:

on utilisant dans notre projet notre modèleChapitre I # (10).. un servo-moteur DC 12v 3000rpm et 200 révolutions ...

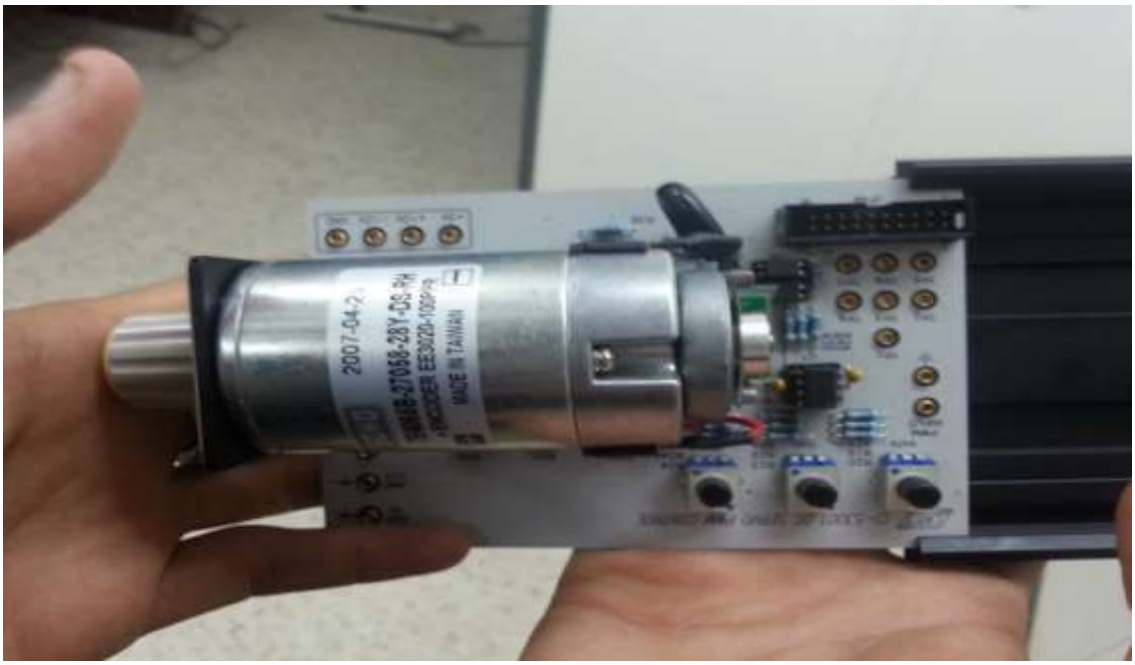


Figure I.35. Servo-motor 27058-28Y-DS-RH-100R

4.1. Les caractéristiques de notre modèle :

Référence du bobinage	Tension nominal	Caractéristique A vide		Caractéristique au meilleur rendement					Caractéristiques Au démarrage		Puissance En pic	Resistance totale	Self inductance	Constante De couple	Force Electro Motrice
	VC/C	mA	Rpm	g-cm	Rpm	mA	%	Watt(s)	g-cm	mA	Watt(s)	Ohm(s)	mH	Kg-Cm/A	V/1000 tours
27058	12	106	3000	185	2538	608	66.12	4.82	1200	3360	3.24	3.57	2.70	0.37	3.80

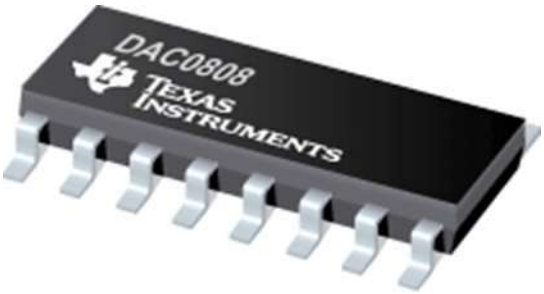
Tableau III.1. Les caractéristiques servo-motor 27058-28Y-DS-RH-100R

5. Plaque d'essais sans soudure



4. les composants électriques :

4.1. DAC 0808



4.2. Ampli ope lf351



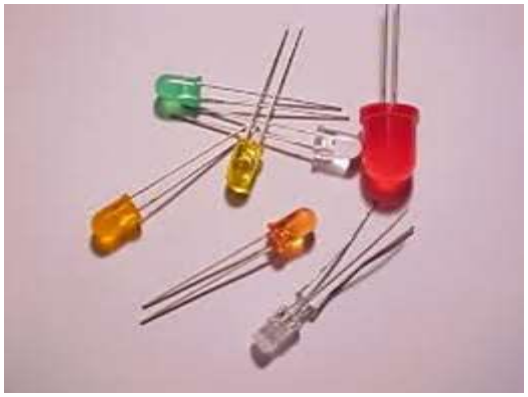
4.3. TP 122 transistors



4.4. Résistance 4.7 et 0.5 kilo-ohm



4.5. Les LEDs



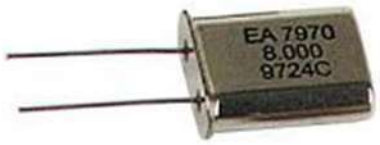
4.6. capacitor 22pf



4.7. Potentiomètre

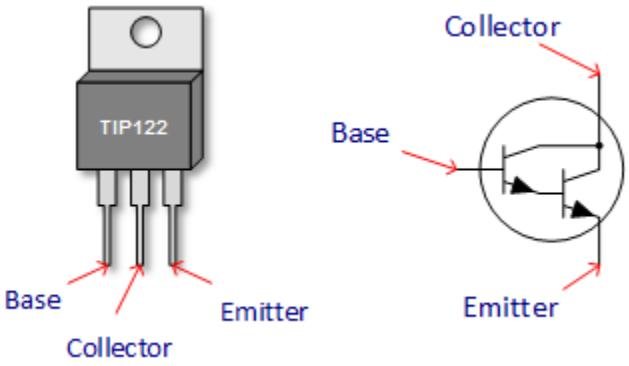


4.8 .quartz électronique



III .5.9.5. Tip 122 transistor:

Le TIP122 est un transistor NPN. Dans ce format, on sait que la base nécessite une tension qui est plus positive que l'émetteur afin de permettre au courant de circuler depuis l'émetteur vers le collecteur. L'emplacement de la base, l'émetteur et le collecteur d'un TIP122 sont présentés dans le schéma ci-dessous.



La chose intéressante à propos de ce transistor est qu'il permet jusqu'à 5 ampères de courant allant de l'émetteur par l'intermédiaire du collecteur et jusqu'à 120 mA va depuis l'émetteur jusqu'à la base.

Tout aussi frais est le fait que vous pouvez avoir autant qu'une différence de 100 volts du collecteur à l'émetteur et autant qu'un volt 100 entre le collecteur et la base.[18].

Bibliographie

[1] Mr. Ramzi SAADI, Mr. SALHI Nassereddine, “réalisation de la carte à un microcontrôleur pour le contrôle de bras manipulateur via un pc ’’, thèse de master, université med khider, Biskra, 2010.

[2] Nadir MAATER, Lembarek BESEKHRIA, ‘‘conception d’une carte de commande de la machine à courant continu par un microcontrôleur pic16F877A ’’, institut nationale de la formation professionnelle CHAHIB BOUKHLOUF MOHAMED BEN EL HADI, Batna, 2009.

[3]http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/chapter/timers/

[4]http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/chapter/core-sfrs/

[5] BRAHIMI Abdelkarim, GUEZOULI Housseyne : ‘‘étude et réalisation d’une carte de commande à base d’un microcontrôleur pic16F877 pour ponts redresseurs triphasés à thyristor’’, thèse de master, université abou-bakr belkaid, tlemcen, 2014

[6] Nadir MAATER, Lembarek BESEKHRIA, ‘‘conception d’une carte de commande de la machine à courant continu par un microcontrôleur pic16F877A ’’, institut nationale de la formation professionnelle CHAHIB BOUKHLOUF MOHAMED BEN EL HADI, Batna, 2009.

[7] Jean-Paul LOUIS, Bernard MULTON, Yvan BONNASIEUX, Michel LAVABRE, ‘‘commande des machines à courant continu (MCC) à vitesse variable’’, article technique de l’ingénieur, Paris, 2002.

[8] Pascal BIGRAS : GPA-783, asservissement numérique en temps réel, école de technologie supérieur, 2011.

[9] http://www.ni.com.

[10] BOUICHE Hachemi, BRAHIMI Mohamed, ‘‘commande PID d’un moteur à courant continu ’’, mini projet, université abderrahmane Mira, Bejaya, 2010.

[11] Mohamed ARHOUJDAM, ‘‘mise en œuvre d’une régulation de vitesse d’un MCC’’, université Mohamed V souissi, 2013.